L Number	Hits	Search Text	DB	Time stamp
1	6943	classif$9 same filter$1	USPAT; EPO; DERWENT; USOCR	2004/01/16 08:57
2	41	barzilai.inv.	USPAT; EPO; DERWENT; USOCR	2004/01/16 08:57
3	1	(classif$9 same filter$1) and barzilai.inv.	USPAT; EPO; DERWENT; USOCR	2004/01/16 08:57
4	420945	(classif$9 same filter$1) samd dynamic$4	USPAT; EPO; DERWENT; USOCR	2004/01/16 08:57
5	152	(classif$9 same filter$1) same dynamic$4	USPAT; EPO; DERWENT; USOCR	2004/01/16 08:58
6	432264	filter$3 same (expir$8 or delet$4 or remov$4 or creat$4)	USPAT; EPO; DERWENT; USOCR	2004/01/16 08:58
7	11494	classif$9 same (packet$1 or message$1 or frame$1 or datagram$1)	USPAT; EPO; DERWENT; USOCR	2004/01/16 08:59
8	12709	admission same (polic$3 or regulat$4 or rule$1 or criteria)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:00
9	70379	(traffic or packet$1 or frame$1 or datagram$1) same (filter$1 or shaping)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:06
10	15	((traffic or packet$1 or frame$1 or datagram$1) same (filter$1 or shaping)) and (admission same (polic$3 or regulat$4 or rule$1 or criteria)) and (classif$9 same (packet$1 or message$1 or frame$1 or datagram$1)) and (filter$3 same (expir$8 or delet$4 or remov$4 or creat$4)) and (classif$9 same filter$1)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:02
11	1093	terrell.inv.	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:02
12	2	terrell.inv. and (nortel.asn. or (bay adj network$1.asn.))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:03
13	5748	(nortel.asn. or (bay adj network$1.asn.))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:03
14	10	((nortel.asn. or (bay adj network$1.asn.))) and (classif$9 same filter$1)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:04
15	8	("5313455" | "5463620" | "5781532" | "6104700" | "6167027" | "6188698" | "6222844" | "6381649").PN.	USPAT	2004/01/16 09:05
16	581148	(traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:09

17	7	("5546390" | "5873078" | "6029170" | "6223174" | "6233574" | "6298340" | "6396842").PN.	USPAT	2004/01/16 09:07
18	11	("5243596" | "5406322" | "5509006" | "5721920" | "5862335" | "5890217" | "6104696" | "6154446" | "6157955" | "6167047" | "6240452").PN.	USPAT	2004/01/16 09:07
19	7	6279035.URPN.	USPAT	2004/01/16 09:08
20	7	("5600820" | "5828844" | "5878043" | "5892924" | "5920705" | "5926459" | "5949786").PN.	USPAT	2004/01/16 09:09
21	0	(("5600820" | "5828844" | "5878043" | "5892924" | "5920705" | "5926459" | "5949786").PN.) and (classif$9 same filter$1)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:09
22	3	(("5600820" | "5828844" | "5878043" | "5892924" | "5920705" | "5926459" | "5949786").PN.) and (classif$9 same (packet$1 or message$1 or frame$1 or datagram$1))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:10
23	2	((("5600820" | "5828844" | "5878043" | "5892924" | "5920705" | "5926459" | "5949786").PN.) and (classif$9 same (packet$1 or message$1 or frame$1 or datagram$1))) and filter$3	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:11
24	759	(filter$3 same (expir$8 or delet$4 or remov$4 or creat$4)) and (classif$9 same filter$1) and ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:11
25	30	((filter$3 same (expir$8 or delet$4 or remov$4 or creat$4)) and (classif$9 same filter$1) and ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy))) and (admission same (polic$3 or regulat$4 or rule$1 or criteria))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:16
26	34	6167445.URPN.	USPAT	2004/01/16 09:15
27	13	6031841.URPN.	USPAT	2004/01/16 09:16
28	152	((classif$9 same filter$1) same dynamic$4) and (classif$9 same filter$1)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:16
29	46	((classif$9 same filter$1) same dynamic$4) and ((traffic or packet$1 or frame$1 or datagram$1) same (filter$1 or shaping))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:16
30	22	(((classif$9 same filter$1) same dynamic$4) and ((traffic or packet$1 or frame$1 or datagram$1) same (filter$1 or shaping))) and @ad<19981228	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:18
31	22387	dynamic$4 same filter$3	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:19
32	1138	(dynamic$4 same filter$3) same ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:19
33	252	((dynamic$4 same filter$3) same ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy))) and (709/$.ccls. or 370/$.ccls.)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:26

34	150	(((dynamic$4 same filter$3) same ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy))) and (709/$.ccls. or 370/$.ccls.)) and @ad<19981228	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:20
35	30	((dynamic$4 same filter$3) same ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy))) and (admission same (polic$3 or regulat$4 or rule$1 or criteria))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:21
36	4447	creat$4 near5 dynamic$4 near5 creat$4	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:21
37	127	(creat$4 near5 dynamic$4 near5 creat$4) same filter$3	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:21
38	80	((creat$4 near5 dynamic$4 near5 creat$4) same filter$3) and @ad<19981228	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:26
39	521	(classif$9 same filter$1) and ((classif$9 same (packet$1 or message$1 or frame$1 or datagram$1)) same ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy)))	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:26
40	93	((classif$9 same filter$1) and ((classif$9 same (packet$1 or message$1 or frame$1 or datagram$1)) same ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy)))) and (709/$.ccls. or 370/$.ccls.)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:28
41	46	(((classif$9 same filter$1) and ((classif$9 same (packet$1 or message$1 or frame$1 or datagram$1)) same ((traffic or packet$1 or frame$1 or datagram$1 or admission) same (filter$1 or shaping or control$4 or policy)))) and (709/$.ccls. or 370/$.ccls.)) and @ad<19981228	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:28
42	591	(classif$9 near10 packet$1).ti. or (classif$9 near10 packet$1).ab.	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:28
43	68	((classif$9 near10 packet$1).ti. or (classif$9 near10 packet$1).ab.) and (709/$.ccls. or 370/$.ccls.)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:56
44	31	(((classif$9 near10 packet$1).ti. or (classif$9 near10 packet$1).ab.) and (709/$.ccls. or 370/$.ccls.)) and @ad<19981228	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:33
45	12	6260072.URPN.	USPAT	2004/01/16 09:31
46	37	("4556972" | "5121383" | "5253248" | "5313454" | "5377327" | "5412654" | "5467345" | "5491801" | "5495479" | "5506847" | "5557607" | "5570346" | "5583861" | "5583862" | "5594734" | "5600630" | "5666360" | "5671222" | "5671445" | "5699361" | "5740164" | "5790536" | "5802278" | "5805816" | "5812526" | "5828844" | "5844887" | "5905712" | "5910942" | "5920566" | "5920568" | "5930254" | "5940372" | "5963546" | "5970232" | "5996021" | "6016306").PN.	USPAT	2004/01/16 09:32

47	2	5848233.pn.	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:55
48	23806	filter$3 same day	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:56
49	103	(filter$3 same day) same (classif$9 same filter$1)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:56
50	3	((filter$3 same day) same (classif$9 same filter$1)) and (709/$.ccls. or 370/$.ccls.)	USPAT; EPO; DERWENT; USOCR	2004/01/16 09:56

US006643260B1

(12) **United States Patent**
Kloth et al.

(10) **Patent No.: US 6,643,260 B1**
(45) **Date of Patent: Nov. 4, 2003**

(54) **METHOD AND APPARATUS FOR IMPLEMENTING A QUALITY OF SERVICE POLICY IN A DATA COMMUNICATIONS NETWORK**

(75) Inventors: **Raymond J. Kloth**, Mountain View, CA (US); **Thomas J. Edsall**, Cupertino, CA (US); **Michael Fine**, San Francisco, CA (US); **Dinesh G. Dutt**, Santa Clara, CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/213,105**

(22) Filed: **Dec. 18, 1998**

(51) **Int. Cl.**[7] **G01R 31/08**; G06F 11/00; G08C 15/00; H04J 1/16; H04J 3/14; H04L 1/00; H04L 12/26; H04L 12/28; H04L 12/56

(52) **U.S. Cl.** **370/235**; 370/412

(58) **Field of Search** 370/229, 235, 370/236, 395.3, 395.31, 395.4, 412, 235.1; 709/223, 226, 229, 238, 239, 240, 242

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,499,576	A	2/1985	Fraser	370/412
4,769,810	A	9/1988	Eckberg, Jr. et al.	370/232
4,769,811	A	9/1988	Eckberg, Jr. et al.	370/236
5,224,099	A	6/1993	Corbalis et al.	370/412
5,253,251	A	10/1993	Aramaki	370/394
5,280,470	A	1/1994	Buhrke et al.	370/232
5,303,237	A	4/1994	Bergman et al.	370/418
5,313,454	A	5/1994	Bustini et al.	370/231

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

WO	99/53408	10/1999	G06F/15/16

OTHER PUBLICATIONS

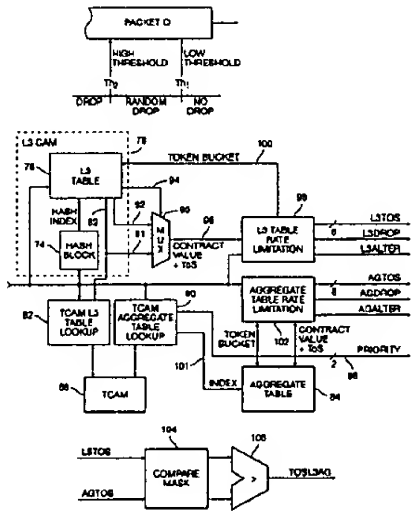
Zhang, et al., "Rate-Controlled Static-Priority Queueing", 1993, IEEE, pp. 227-236.

Primary Examiner—Seema S. Rao
Assistant Examiner—Kevin C. Harper
(74) *Attorney, Agent, or Firm*—Thelen Reid & Priest LLP; David B. Ritchie

(57) **ABSTRACT**

A content addressable memory (CAM or L3 Table) contains flow information for each active flow of packets passing through a given node of a data communications network. The CAM has associated with each entry (corresponding to each active flow) a packet counter, a byte counter, a token bucket and a contract value. Each flow is assigned one of a plurality of output queues and optionally at least one output threshold value. A token bucket algorithm is employed on each flow to determine whether packets from that flow exceed the contract value. Such packets may be dropped or optimally modified to reflect an alternate output queue and/or alternate threshold before being sent to the selected output queue for transmission from the node. In another aspect an access control list CAM (ACLCAM) contains masked flow information. The ACLCAM provides an index to internal token bucket counters and preconfigured contract values of an aggregate flow table which becomes affected by the packet statistics. In this way flows are aggregated for assignment of output queues and thresholds, possible dropping and possible modification of packets. In another aspect the CAM contains active flow information, the ACLCAM and the aggregate flow table are combined in one system and used to produce in parallel a pair of traffic rate limiting and prioritizing decisions for each packet. The two results are then resolved to yield a single result.

24 Claims, 7 Drawing Sheets



U.S. PATENT DOCUMENTS

5,313,579	A *	5/1994	Chao	370/230.1
5,317,562	A	5/1994	Nardin et al.	370/428
5,359,592	A	10/1994	Corbalis et al.	370/233
5,408,472	A	4/1995	Hluchyj et al.	370/416
5,467,349	A *	11/1995	Huey et al.	370/397
5,485,455	A	1/1996	Dobbins et al.	370/255
5,497,371	A	3/1996	Ellis et al.	370/394
5,502,725	A	3/1996	Pohjakallio	370/337
5,515,363	A *	5/1996	Ben-Nun et al.	370/232
5,570,360	A	10/1996	Klausmeier et al.	370/232
5,570,361	A	10/1996	Norizuki et al.	370/392
5,598,581	A	1/1997	Daines et al.	370/401
5,610,910	A	3/1997	Focsaneanu et al.	370/351
5,666,361	A *	9/1997	Aznar et al.	370/392
5,694,554	A *	12/1997	Kawabata et al.	370/412
5,699,521	A	12/1997	Iizuka et al.	370/455
5,712,854	A	1/1998	Dieudonne et al.	370/536
5,734,654	A	3/1998	Shirai et al.	370/396
5,737,635	A	4/1998	Daines et al.	709/232
5,765,032	A	6/1998	Valizadeh	370/235
5,778,182	A	7/1998	Cathey et al.	709/219
5,793,978	A	8/1998	Fowler	709/201
5,805,595	A	9/1998	Sharper et al.	370/442
5,828,653	A *	10/1998	Goss	370/230
5,835,494	A	11/1998	Hughes et al.	370/232
5,835,727	A	11/1998	Wong et al.	709/238
5,838,683	A	11/1998	Corley et al.	370/408
5,922,051	A	7/1999	Sidey	709/223
5,926,458	A	7/1999	Yin	370/230
5,949,784	A *	9/1999	Sodder	370/397
5,959,990	A	9/1999	Frantz et al.	370/392
5,970,477	A	10/1999	Roden	379/112.01
6,031,820	A *	2/2000	Kawasaki et al.	370/230
6,035,281	A	3/2000	Crosskey et al.	705/14
6,047,326	A *	4/2000	Kilkki	370/232
6,091,708	A *	7/2000	Matsunuma	370/233
6,119,160	A	9/2000	Zhang et al.	709/224
6,122,252	A *	9/2000	Aimoto et al.	370/235
6,157,613	A *	12/2000	Watanabe et al.	370/229
6,167,445	A *	12/2000	Gai et al.	709/220
6,275,494	B1 *	8/2001	Endo et al.	370/395.52
6,335,932	B2 *	1/2002	Kadambi et al.	370/391
2002/0085496	A1 *	7/2002	Jamp et al.	370/235

* cited by examiner

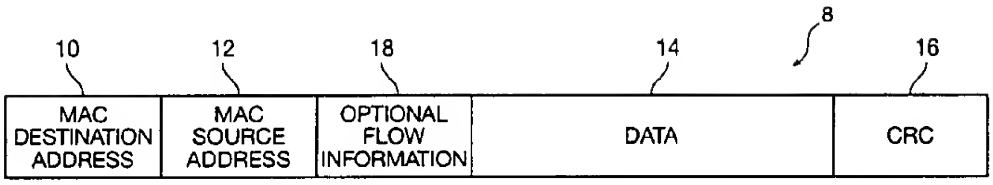

8
10
12
18
14
16
MAC DESTINATION ADDRESS
MAC SOURCE ADDRESS
OPTIONAL FLOW INFORMATION
DATA
CRC

FIG. 1A

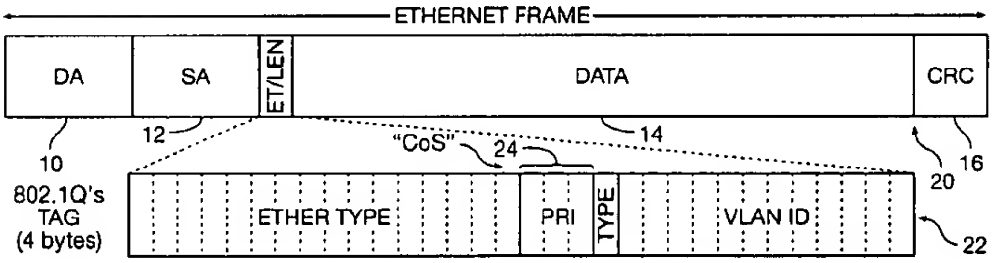

ETHERNET FRAME
DA
SA
ET/LEN
DATA
CRC
10
12
14
16
20
"CoS"
24
802.1Q's TAG (4 bytes)
ETHER TYPE
PRI
TYPE
VLAN ID
22

FIG. 1B

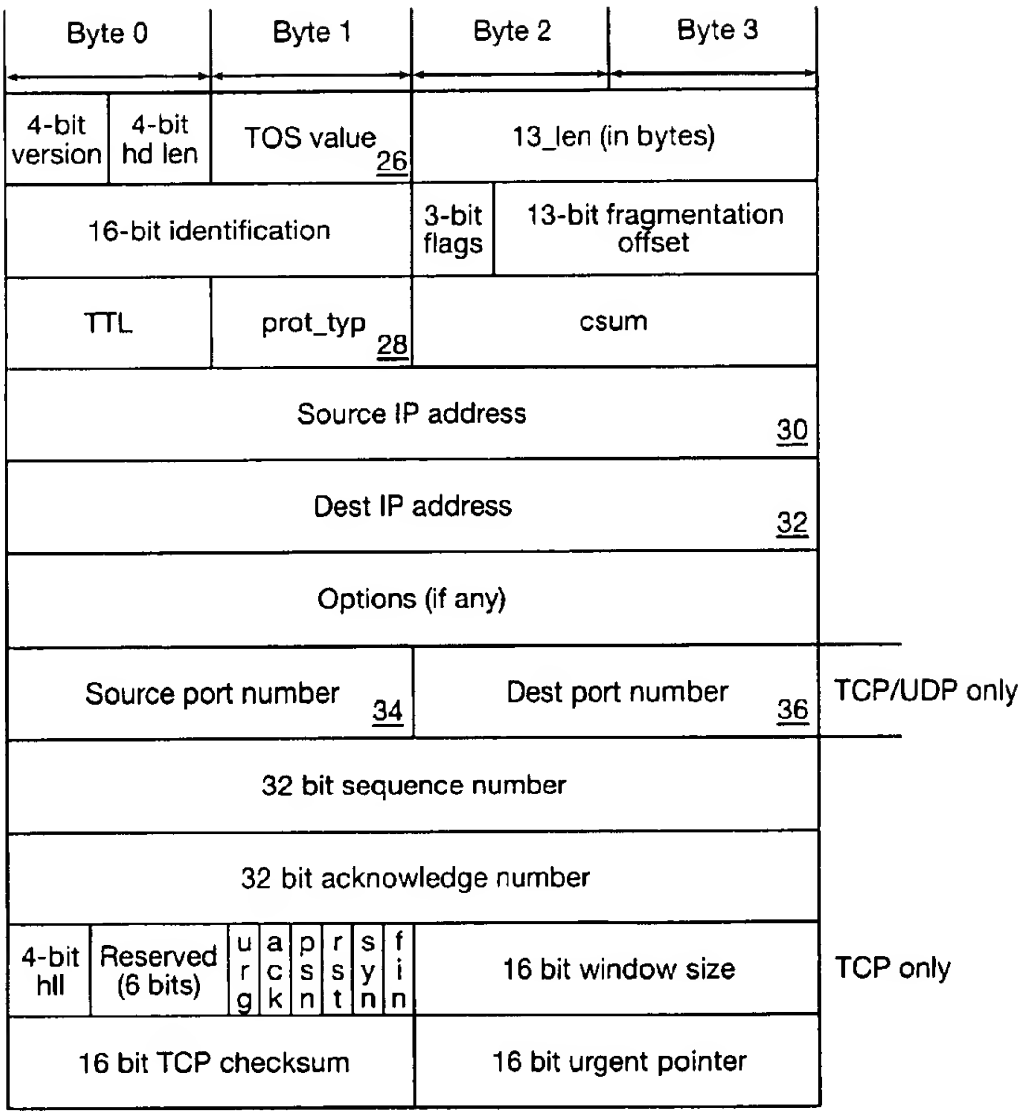

Byte 0
Byte 1
Byte 2
Byte 3
4-bit version
4-bit hd len
TOS value 26
13_len (in bytes)
16-bit identification
3-bit flags
13-bit fragmentation offset
TTL
prot_typ 28
csum
Source IP address 30
Dest IP address 32
Options (if any)
Source port number 34
Dest port number 36
TCP/UDP only
32 bit sequence number
32 bit acknowledge number
4-bit hll
Reserved (6 bits)
urg
ack
psn
rst
syn
fin
16 bit window size
TCP only
16 bit TCP checksum
16 bit urgent pointer

FIG. 1C

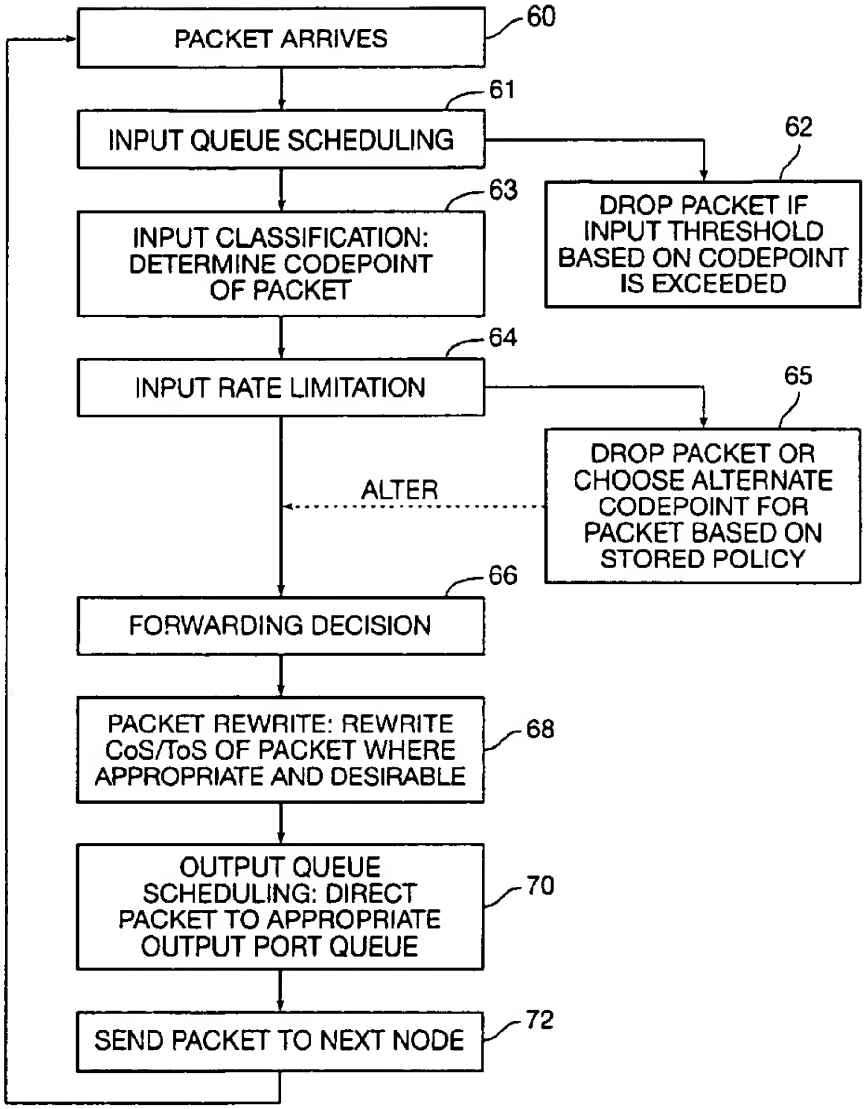

PACKET ARRIVES
60
INPUT QUEUE SCHEDULING
61
62
DROP PACKET IF INPUT THRESHOLD BASED ON CODEPOINT IS EXCEEDED
INPUT CLASSIFICATION: DETERMINE CODEPOINT OF PACKET
63
INPUT RATE LIMITATION
64
65
DROP PACKET OR CHOOSE ALTERNATE CODEPOINT FOR PACKET BASED ON STORED POLICY
ALTER
FORWARDING DECISION
66
PACKET REWRITE: REWRITE CoS/ToS OF PACKET WHERE APPROPRIATE AND DESIRABLE
68
OUTPUT QUEUE SCHEDULING: DIRECT PACKET TO APPROPRIATE OUTPUT PORT QUEUE
70
SEND PACKET TO NEXT NODE
72

FIG. 2

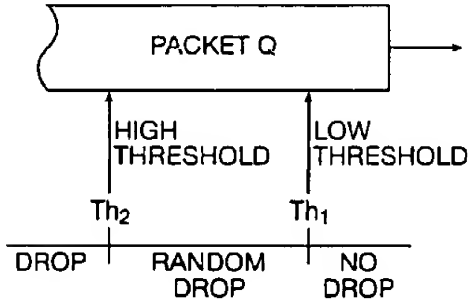

PACKET Q
HIGH THRESHOLD
LOW THRESHOLD
Th2
Th1
DROP
RANDOM DROP
NO DROP

FIG. 3

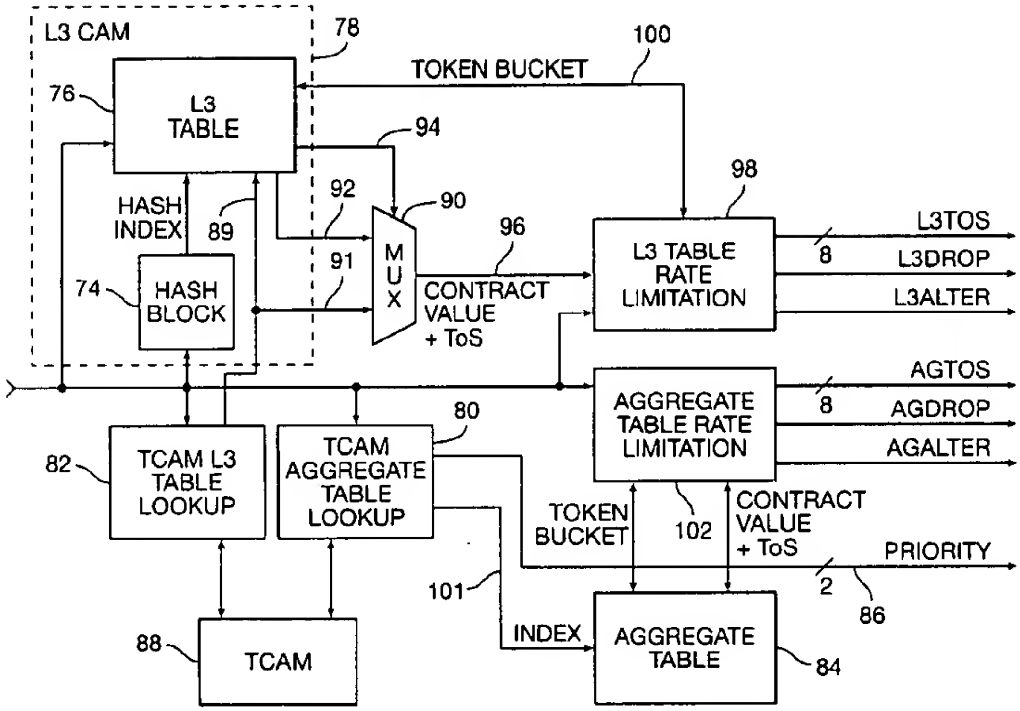

L3 CAM
78
100
TOKEN BUCKET
76
L3 TABLE
94
98
92
90
HASH INDEX
89
96
L3TOS
8
L3DROP
L3ALTER
74
HASH BLOCK
91
M U X
CONTRACT VALUE + ToS
L3 TABLE RATE LIMITATION
AGTOS
8
AGDROP
AGALTER
AGGREGATE TABLE RATE LIMITATION
80
82
TCAM L3 TABLE LOOKUP
TCAM AGGREGATE TABLE LOOKUP
TOKEN BUCKET
102
CONTRACT VALUE + ToS
PRIORITY
2
86
101
INDEX
AGGREGATE TABLE
84
88
TCAM

FIG. 4A

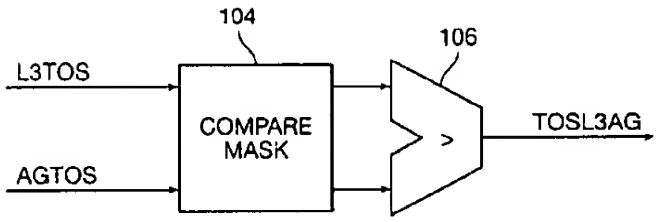

104
106
L3TOS
COMPARE MASK
>
TOSL3AG
AGTOS

FIG. 4B

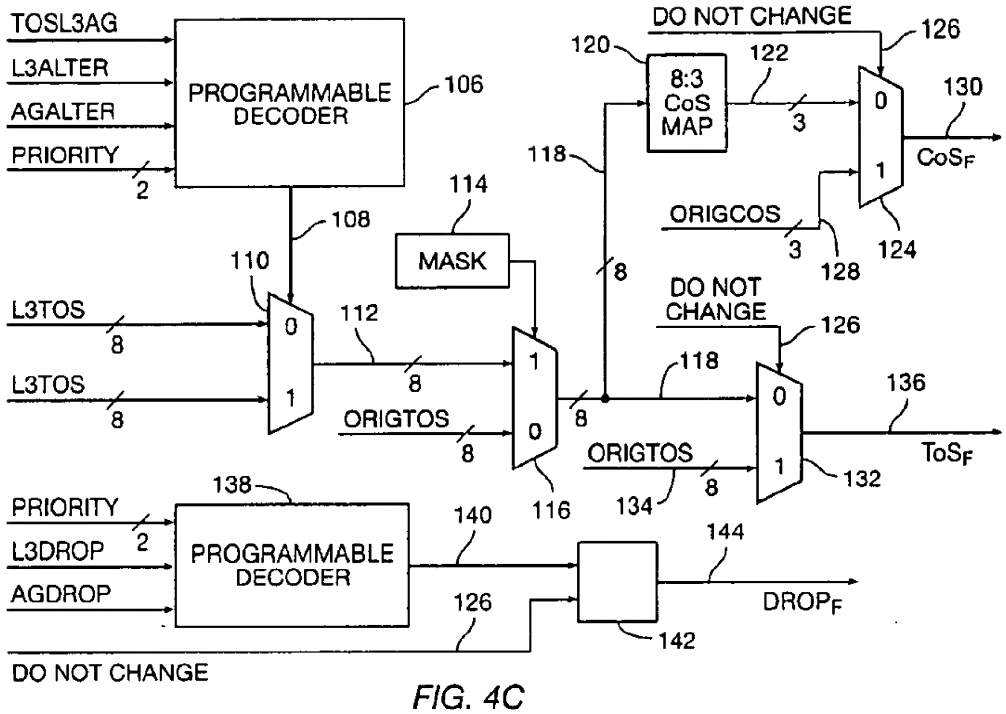

TOSL3AG
L3ALTER
AGALTER
PRIORITY
2
PROGRAMMABLE DECODER
106
108
110
L3TOS
8
L3TOS
8
0
1
112
8
114
MASK
1
0
ORIGTOS
8
116
8
118
120
8:3 CoS MAP
122
3
DO NOT CHANGE
126
0
1
130
CoSF
ORIGCOS
3
128
124
DO NOT CHANGE
126
118
0
1
136
ToSF
ORIGTOS
8
134
132
PRIORITY
2
L3DROP
AGDROP
138
PROGRAMMABLE DECODER
140
144
DROPF
126
142
DO NOT CHANGE

FIG. 4C

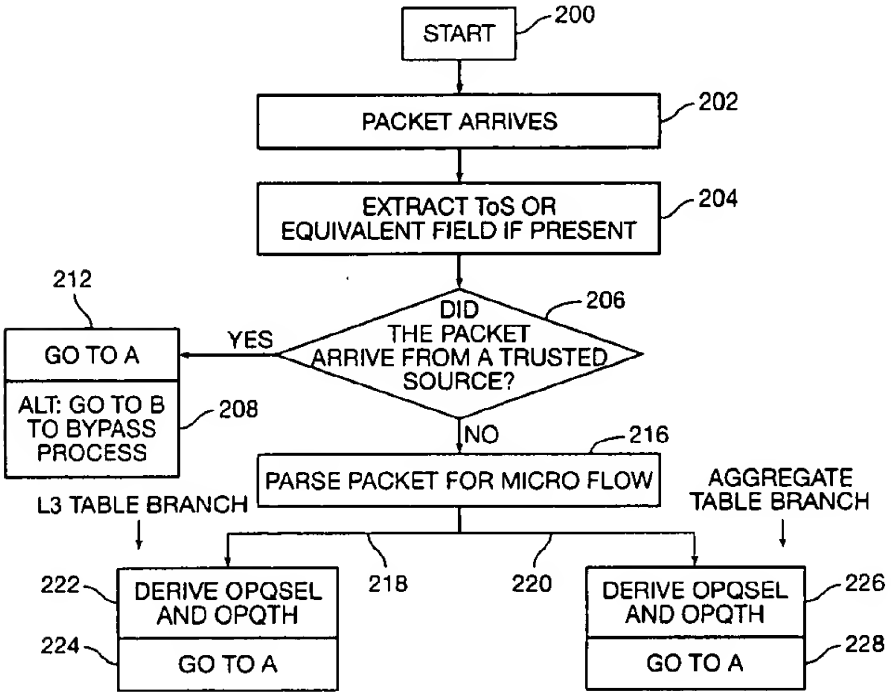

START
200
PACKET ARRIVES
202
EXTRACT ToS OR EQUIVALENT FIELD IF PRESENT
204
212
DID THE PACKET ARRIVE FROM A TRUSTED SOURCE?
206
YES
GO TO A
ALT: GO TO B TO BYPASS PROCESS
208
NO
216
PARSE PACKET FOR MICRO FLOW
L3 TABLE BRANCH
AGGREGATE TABLE BRANCH
222
DERIVE OPQSEL AND OPQTH
224
GO TO A
218
220
DERIVE OPQSEL AND OPQTH
226
GO TO A
228

FIG. 5A

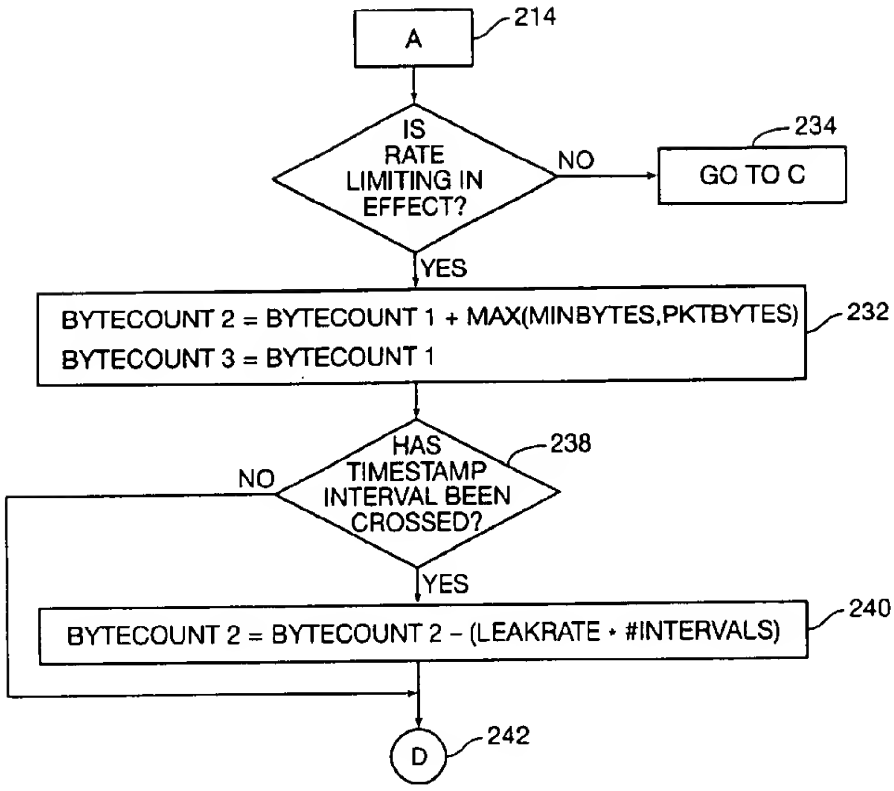

A
214
IS RATE LIMITING IN EFFECT?
NO
234
GO TO C
YES
BYTECOUNT 2 = BYTECOUNT 1 + MAX(MINBYTES,PKTBYTES)
BYTECOUNT 3 = BYTECOUNT 1
232
HAS TIMESTAMP INTERVAL BEEN CROSSED?
238
NO
YES
BYTECOUNT 2 = BYTECOUNT 2 – (LEAKRATE * #INTERVALS)
240
D
242

FIG. 5B

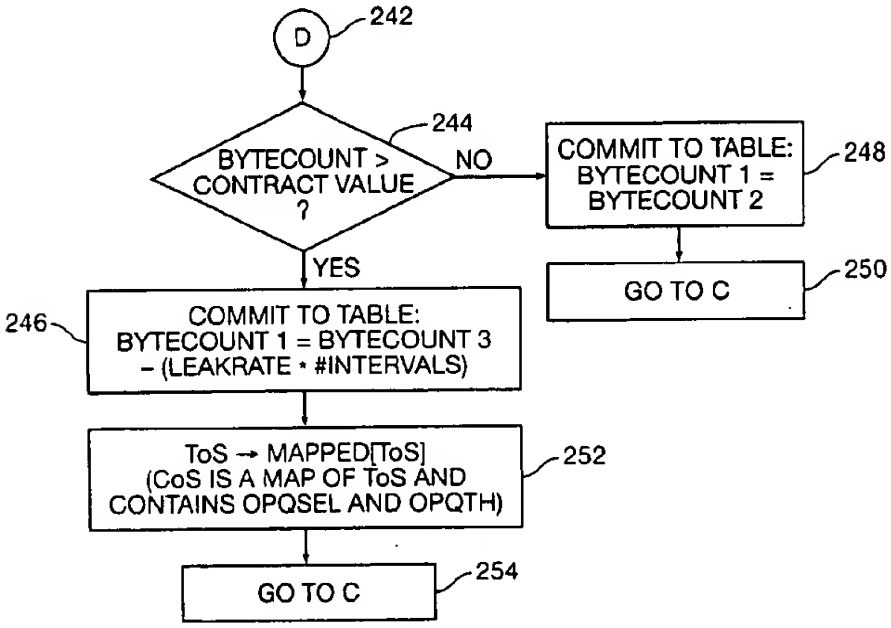

D
242
BYTECOUNT > CONTRACT VALUE ?
244
NO
COMMIT TO TABLE: BYTECOUNT 1 = BYTECOUNT 2
248
GO TO C
250
YES
246
COMMIT TO TABLE: BYTECOUNT 1 = BYTECOUNT 3 – (LEAKRATE * #INTERVALS)
ToS → MAPPED[ToS] (CoS IS A MAP OF ToS AND CONTAINS OPQSEL AND OPQTH)
252
GO TO C
254

FIG. 5C

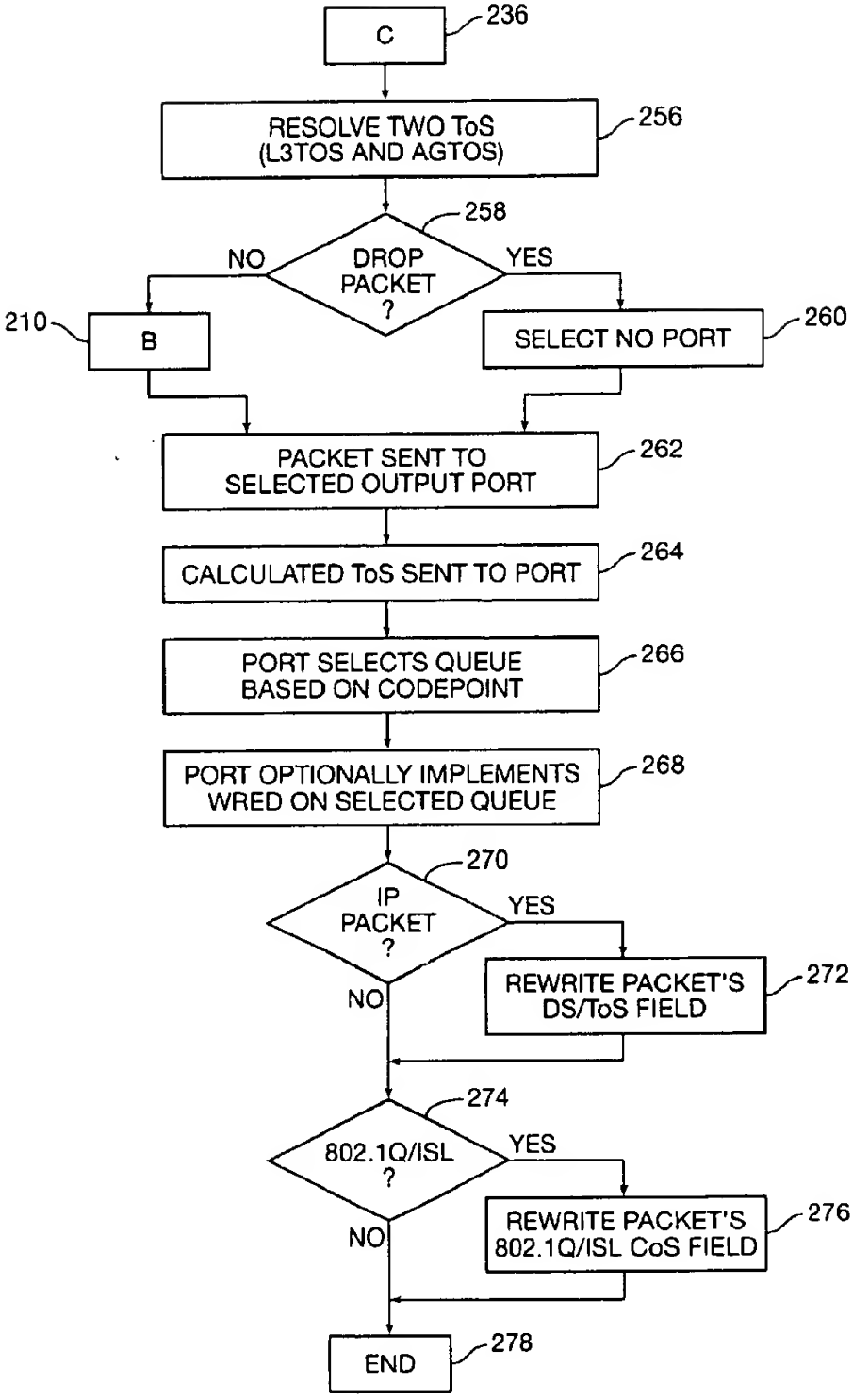

C
236
RESOLVE TWO ToS (L3TOS AND AGTOS)
256
258
DROP PACKET ?
NO
YES
210
B
SELECT NO PORT
260
PACKET SENT TO SELECTED OUTPUT PORT
262
CALCULATED ToS SENT TO PORT
264
PORT SELECTS QUEUE BASED ON CODEPOINT
266
PORT OPTIONALLY IMPLEMENTS WRED ON SELECTED QUEUE
268
270
IP PACKET ?
YES
NO
REWRITE PACKET'S DS/ToS FIELD
272
274
802.1Q/ISL ?
YES
NO
REWRITE PACKET'S 802.1Q/ISL CoS FIELD
276
END
278

FIG. 5D

METHOD AND APPARATUS FOR IMPLEMENTING A QUALITY OF SERVICE POLICY IN A DATA COMMUNICATIONS NETWORK

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of data communications networks. More particularly, this invention relates to a method and apparatus for implementing a quality of service (QoS) policy in a data communications network so as to thereby prioritize network traffic into a plurality of service levels and provide preferential treatment of different classes of data traffic on the data communications network. A number of priority levels may be implemented in accordance with the invention.

2. Background

This invention relates to switched packet data communications networks. There are a number of different packet types which are used in modern switched packet data communications networks.

FIG. **1A** depicts a generic packet **8** using Layer **2** encapsulation. A number of different Layer **2** encapsulation protocols are recognized. Each may include a MAC (media access control) destination address **10** and a MAC source address **12**. The data **14** may include Layer **3** encapsulated packet information. A CRC (cyclic redundancy check) **16** may also be provided at the end of the Layer **2** encapsulation. The unlabeled block **18** may include an Ethernet type for Ethernet V 2.0 (ARPA) packets. The Ethernet type may include IPv4 (IP), IPX, AppleTalk, DEC Net, Vines IP/Vines Echo, XNS, ARP or RARP. Other known encapsulations include SAP, SAP1, SNAP and the like. The meaning of the bits in and the size of block **18** differs among the different encapsulation protocols. This information is sometimes referred to as the Layer **2** Flow Information.

One special case of Layer **2** encapsulation is the IEEE 802.1q frame shown schematically in FIG. **1B**. The IEEE 802.1q frame (or packet) **20** has a MAC Destination Address ("DA") **10**, MAC Source Address ("SA") **12**, Data Portion **14** and CRC **16**. In addition, within block **18** is the IEEE 802.1q "tag" **22** which includes, among other items, a block of three priority bits **24**. These three bits are also known as a "Class of Service" or "CoS" field.

FIG. **1C** depicts the Layer **3** and Layer **4** structure of a typical IP packet The IP packet format will be detailed here by way of example because it is presently one of the most common Layer **3** packet types. The fields of importance to this disclosure are the "ToS value" or type of service **26** which is a preferably 8-bit field also known as the Differentiated Service ("DS") field, "prot-typ" or IP protocol type **28** (typically either TCP (transmission control protocol) or UDP (user datagram protocol)), the Source IP address **30** (usually the IP address of the originating station), the Destination IP address **32** (usually the IP address of the ultimate destination station), the Layer **4** source port number **34** (available for TCP and UDP packets only) and the Layer **4** destination port number **36** (available for TCP and UDP packets only). The Layer **3** flow information includes the information before the source port number **34**. The Layer **4** flow information includes the Source and Destination ports **34, 36**. The Layer **4** flow information may be used to identify a particular packet flow as being the product of (source port) or directed to (destination port) a particular application. The ToS and CoS fields are used by routers of the data communications network to provide priority/delay/dropping services.

As the use of data communications networks increases worldwide, congestion of those networks has become a problem. A given data communications network, a given node on a data communications network, or a given link connecting two nodes has a certain capacity to pass data packets and that capacity cannot be exceeded. When data traffic on the data communications network becomes heavy enough that one can anticipate congestion problems, it is desirable to implement a "Quality of Service" or QoS policy so as to give priority to certain types of traffic and restrict the flow of other types of traffic, thus assuring that critical communications are able to pass through the data communications network, albeit at the expense of less critical communications.

One of the problems that network devices face in implementing quality of service solutions is in identifying and grouping transmissions to be given preferential treatment or to be restricted, that is, to prioritize the traffic in accordance with the Quality of Service policy established for the network. This becomes especially critical as bandwidth increases substantially over certain links while other links remain relatively slow resulting in traffic speed mismatches which, in turn, cause bottlenecks to data traffic over the relatively slow links. Such groupings must be consistently applied to traffic and must be applied at the rate that the traffic is passing without introducing additional delays or bottlenecks. Such groupings may be, for example, by protocol type, by destination IP address, by source IP address, by destination/source IP address pair, by source port and/or destination port (Layer **4**), and the like.

Routers have, in the past, kept packet counts and rate limited packets in software, but router software has not scaled to the level of being able to process millions of packets per second through a node, providing the basic routing functions that they are required to provide and being able to also provide the rate limitation function.

One approach to identifying and grouping transmissions is for the host to categorize packets by use of the L2 CoS field, L3 ToS field or both. The primary disadvantage of this approach is that it removes control from the system administrator and requires one to trust the end stations to the communication to properly implement the QoS policy. In some cases this trust cannot be justified. In addition, an end station only sees its own packets and therefore is unaware of the overall resource requirements within the data communications network and cannot make allowances for these requirements.

Accordingly, a Quality of Service policy controlled by a network system administrator is needed together with a mechanism for applying it at the full data rate of the data communications network.

SUMMARY OF THE INVENTION

In a first aspect of the invention a content addressable memory (CAM or L3 Table) contains flow information for each active flow of packets passing through a given node of a data communications network. The CAM has associated with each entry (corresponding to each active flow) a packet counter, a number of bytes seen counter, a token bucket and a contract value or committed access rate. Each flow is assigned one of a plurality of output queues and optionally at least one output queue threshold value. A token bucket algorithm is employed on each flow to determine whether packets from that flow exceed a committed access rate. Such packets may be dropped or optionally modified to reflect an alternate output queue and/or alternate output queue thresh-

old value before being sent to the selected output queue for transmission from the node.

In a second aspect of the invention an access control list CAM (ACLCAM) contains masked flow information such as, for example, all or portions of IP source and/or destination addresses, protocol types and the like. The ACLCAM provides single clock cycle accesses when performing lookups for each packet. The ACLCAM provides an N-bit index value in response to QoS lookups based upon the best match for the current packet.

The best match is order dependent for the entry in the ACLCAM and may represent any fields in the packet upon which the administrator of the data communications network wishes to base traffic rate limiting and prioritizing decisions. A plurality of ACLCAM entries can yield the same N-bit index value. The N-bit ACLCAM index selects one of 2^N internal counters and associated preconfigured contract values, which become affected by the packet statistics. A token bucket algorithm is employed on these counters as discussed above.

The ACL CAM may also be used to determine the QoS parameters for new entries in the L3 Table as they are created. In addition, it is used to select an entry in the aggregate flow table described below.

In a third aspect of the invention, an aggregate flow table contains information specifying plural flows—for example all traffic between x and y regardless of type, all traffic to x of a certain type, all traffic from anyone of a certain type, and the like. These specifications may specify more than one flow. This is possible because each entry has a corresponding flow mask. This is different from the L3 Table which may identify certain specific flows only, i.e., all traffic of protocol type HTTP from x to y. Since the entire L3 Table operates with a single flow mask, each entry will have identical specificity, thus, there could be multiple entries for traffic from x to y if such traffic includes multiple protocol types and the flow mask does not mask the protocol type, for example.

In a fourth aspect of the invention, the CAM, an aggregate flow table and the ACLCAM are combined in one system and used to produce, in parallel, a pair of traffic rate limiting and prioritizing decisions for each packet. The two results are then resolved (if in conflict) to yield a single result which is acted upon. The result is to modify or not modify the packet's CoS and/or ToS (or other) fields and to drop or pass the packet onto the next node of the data communications network.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A is a diagram showing the structure of a typical Ethernet packet.

FIG. 1B is a diagram showing the structure of a typical Ethernet packet including the IEEE 802.1q tag.

FIG. 1C is a diagram showing the structure of a Layer 3 IP packet.

FIG. 2 is a block diagram showing the implementation of a Quality of Service policy.

FIG. 3 is a diagram showing the functional operation of an output queue implementing threshold-based dropping.

FIGS. 4A, 4B and 4C are a system block diagram of an apparatus in accordance with a presently preferred embodiment of the present invention.

FIGS. 5A, 5B, 5C and 5D are a flow diagram of packet processing in accordance with a presently preferred embodiment of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Those of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not intended to be in any way limiting. Other embodiments of the invention will readily suggest themselves to such skilled persons from an examination of the within disclosure.

In accordance with a presently preferred embodiment of the present invention, the components, process steps, and/or data structures are implemented using ASIC technology. This implementation is not intended to be limiting in any way. Different implementations may be used and may include various types of operating systems, computing platforms, and/or computer programs. In addition, those of ordinary skill in the art will readily recognize that devices of a more general purpose nature, such as hardwired devices, devices relying on FPGA technology, and the like, may also be used without departing from the scope and spirit of the inventive concepts disclosed herewith.

Introduction

Unless data communications networks are provisioned with large excess bandwidth, there may be times when the offered load at a given link will be greater than the capacity of that link. This results in greater than average packet delay or even dropping packets in excess of the link capacity. While this may be acceptable to many applications, it can effectively result in loss of service for others. Furthermore, user policies may dictate that some traffic is more important than other traffic and should, therefore, be given preferential treatment in these congestion situations. Even in uncongested conditions, it may be desirable to give preferential treatment to traffic with more stringent real time requirements (such as voice and video) so as to avoid the delay incurred waiting for the transmission of long packets with less stringent real time requirements.

Providing preferential forwarding treatment to some traffic, perhaps at the expense of other traffic, is referred to by the general term Quality of Service (QoS). QoS comprises four distinct functions. These are (1) Classification; (2) Rate Limitation; (3) Packet Rewrite; and (4) Scheduling.

Classification is the process by which a QoS label is assigned to a packet. The QoS label is represented by a codepoint used internally by the device which determines how the packet is treated as it flows through the device. A codepoint is an integer or other representation representing the QoS classification that the device assigned the packet to. The codepoint also determines the value written into the packet's CoS (for 802.1q packets) and ToS (for IP packets) fields.

CoS means Class of Service. This is the name given to three bits in the Layer 2 header (CoS 24 in FIG. 1B) that indicate the QoS assigned to this packet. These three bits are located in the 802.1q header for 802.1q-encoded packets and in the user field of the ISL (Inter-Switch Link) header for ISL-encapsulated packets. Those of skill in the art will realize that the present invention will work essentially interchangeably with 802.1q—tagged frames and ISL frames as well as any other scheme including QoS encoding.

ToS means Type of Service. It is a preferably 1 byte (8-bit) field in the IP header (ToS 26 in FIG. 1C) that indicates the QoS assigned to an IP packet. Since the ToS field is not available for all packet types, the CoS field is also used. ToS is in the process of being redefined as "Differentiated

Services" (DS). The ToS/DS field can select among up to 256 (2^8) different queues, for example.

Input to the classification function includes user policies that map ACEs to codepoints. ACE means access control entry. It is a filter that is used to identify flows with certain characteristics. It includes fields such as device input and/or output ports, input and/or output VLANS, Layer 2 addresses, Layer 3 addresses, TCP/UDP (Layer 4) port numbers, and the like.

A committed access rate (CAR) is the bandwidth that a specific flow or group of flows will be limited to. The CAR can be enforced by rate limitation (dropping out-of-profile packets under certain levels of congestion in accordance with a selected algorithm) or by shaping.

The result of classification is a codepoint assigned (internal to the device) to the packet. (Depending on the user policies, it may simply be set to the CoS or ToS or other field initially taken from the packet).

There are three ways that user policy can control the classification of a packet.

1. By specifying the codepoint for a port (e.g., a particular hardware port of the device, a device input subnet or a device input VLAN);

2. By specifying the codepoint for packets with a specific MAC destination address in a specific VLAN; and

3. By specifying the codepoint for packets matching a specific ACE.

In accordance with a presently preferred embodiment of the present invention, the algorithm for determining the codepoint of a packet consists of three distinct steps.

First, the packet is classified on the basis of the input port. If the port is a trunk port (carrying traffic from multiple VLANs) and the packet has an ISL or 802.1q header then the classification of the packet is the CoS of the packet. If the port is a trunk port and the packet does not have an ISL or 802.1q header, or the port is an access port (carrying traffic for a single VLAN), then the classification of the packet is set to the CoS value configured on that port. Each port is configured with a single CoS value.

Second, a check is made to see if a CoS has been explicitly configured for the packet's MAC destination address. If it has then the packet is assigned the CoS configured for that address replacing the previously assigned CoS.

Third, a check is made to see if it matches any of the configured ACEs. If the packet matches a configured ACE then the packet is assigned the CoS corresponding to that ACE, replacing the previously assigned value. Once a matching ACE is found, no others are checked. Thus, the order of the checking of the ACEs matters or a mechanism is required to resolve multiple matches.

Rate limitation or traffic restriction is the process by which the switch limits the bandwidth consumed by an individual flow or a group of flows counted together as an "aggregate". Rate limitation consists of two stages. The first stage determines if a given packet is in profile or out of profile. That is, whether or not the packet causes the flow (or aggregation of flows) to exceed its allotted bandwidth (CAR) or not. An "in profile" packet is a packet that does not cause the CAR of the packet's flow to be exceeded. An "out of profile" packet is the converse. The second stage of rate limitation assigns an action to apply to packets that are out of profile. This action may be either to reassign the packet a new codepoint, or to drop the packet. Input to the rate limitation function includes: (1) user policies in terms of ACEs and QoS parameters; (2) device input port (or subnet or VLAN); (3) the codepoint the switch assigned to the packet; and (4) the packet flow including layers 2, 3 and 4. The output is a new codepoint, which may be either the original one or the new one, and a boolean value to indicate whether or not to drop the packet. If the packet is dropped the new codepoint is irrelevant.

Packet rewrite is the process by which the device writes a new CoS 24 (for all IEEE 802.1q packets) and/or ToS 26 (for IP packets only) into the packet. These values are derived from the codepoint, preferably through a conventional mapping function. Input to the rewrite function is the packet's codepoint, the codepoint to CoS mapping and the codepoint to ToS mapping. Other types of packets may employ packet rewrite such as ISL encapsulated packets and the like.

Depending on how the packet is classified, the rewrite function rewrites either the packet CoS 24 or both the CoS 24 and the IP ToS 26. If the packet is classified on the basis of an IP ACE, both the CoS and the ToS are rewritten.

Note that for packets going out an access port or where the packet's VLAN is the native VLAN of a trunk port, the packet may be transmitted without an ISL or 802.1q header. In this case the CoS value is lost once the packet leaves the device. However, the CoS value is still used internally by the device in performing the scheduling function.

Scheduling includes: the process by which the device picks an output queue for the packet, the policy for dropping packets when the queue exceeds some threshold (tail drop (dropping packets while the threshold is exceeded), RED (random early detection), etc.) and the algorithm for servicing the various queues (priority queueing, WRR (weighted round robin), etc.). Input to the scheduling function includes user policies that specify queue and scheduling configuration; user policies that map codepoints to queues; and the codepoint that was the output of the rate limitation function, i.e., the packet's current codepoint. The packet is enqueued on the appropriate queue or (perhaps randomly) dropped if the rate exceeds the CAR for this codepoint.

The processing of the packet is diagrammed at FIG. 2. The first operation after arrival of the packet at block 60 is preferably an input queue scheduling process 61 where packets can be dropped (at reference numeral 62) at the input to the device under congestion conditions if an input threshold based upon the codepoint is exceeded. The next operation is preferably input classification 63 since it is not generally possible to do any of the other functions before the packet has been classified. In classification the codepoint (from which may be derived the ToS and/or CoS) of the packet is determined. The codepoint is determined for all packets even if they are not packets which normally include ToS and/or CoS fields and these ToS/CoS values are used internally to the device.

Immediately after input classification is input rate limitation 64 where at block 65 the packet may be dropped or its codepoint altered based upon stored policies configurable by an administrator of the system. For example, if an out-of-profile packet arrives, it may be dropped, or its codepoint may be altered to make it more likely to be dropped down the line.

Following input rate limitation is a forwarding decision 66. The forwarding decision 66 is not a part of the QoS, but it determines the output port of the device to use which, in this general model, is a parameter to the output queue scheduling process 70 discussed below.

Following this is the Packet Rewrite operation 68 where the CoS and/or ToS or other field of the packet is rewritten if appropriate and desirable.

Next, output queue scheduling 70 is performed and the packet is directed to an appropriate queue of the previously selected output port based upon the codepoint determined for the packet in the classification operation or the packet's subsequently altered codepoint. The threshold for the output queue is also selected here.

Finally, at 72 the packet is sent to the next node in the data communications network.

In an alternative embodiment, the device output port (or subnet or VLAN) could be a parameter to the classification function and, thus, a second classification function and a second rate limitation operation could be applied after the forwarding decision.

Output scheduling depends upon the capabilities of the output port. Most prior art ports are FIFO (first in, first out) ports. Such ports are not capable of output scheduling in accordance with all aspects of the present invention. In accordance with one presently preferred embodiment of the present invention, an output port having two queues each with two configurable WRED (weighted random early detection) classes is used. WRED is well known to those of ordinary skill in the art. Each CoS is mapped as desired to one of those WRED classes. For each class there is preferably a low threshold Th_1 and a high threshold Th_2. The low value Th_1 specifies the average depth of the queue below which packets that map to that threshold's class will not be dropped. The high value Th_2 specifies the average queue depth above which packets will be dropped with probability 1. For average queue depths between the low and high values packets are randomly dropped. This is shown in FIG. 3 for one class. It is possible to set the high and low values for each threshold to be the same or nearly the same. The result is a queue with four thresholds with tail drop or near tail drop performance when a threshold is hit. Tail drop means dropping all packets while the threshold is exceeded and no packets while the threshold is not exceeded.

In accordance with another presently preferred embodiment of the invention, packets are queued for transmission in one of a plurality of output queues. For example, two queues could be used, for example, a high priority queue and a low priority queue, or a voice queue and a data queue. Many queues could also be used to give more range to the priority processing of packets.

In accordance with a presently preferred embodiment of the invention, each queue has a fixed depth or memory capacity. Variable depth queues could be used as will be recognized by those of ordinary skill in the art. Each queue also has associated with it at least one threshold, the value of which is programmable.

As presently preferred, a WRED (weighted random early detection) algorithm may be used to determine the probability of dropping a packet based upon the state of fullness of its queue. For example, in a queue having two thresholds Th_1 and Th_2 (see FIG. 3) for $Th_2 \geqq Th_1$, the more full the queue is, over a period of time, and past a particular threshold such as Th_1, the more likely a packet is to be dropped. The purpose here is to protect the higher priority traffic. Suppose that there is high priority traffic such as traffic used to control and regulate the operation of the communications network. If such traffic could not get through to its destination, the network might fail. Thus it is desirable to set the threshold of other traffic so that it is dropped well before the time that the network becomes so congested that high priority traffic is at risk.

By selecting a relatively low value for Th_1 for the low priority queue, the low priority traffic in the low priority queue will be dropped well before high congestion is experienced on the data communications network. On the other hand, a much higher value for Th_1 is appropriate for mission critical frames—for example: do not drop until the queue is 100% full.

In effect, this system allows for certain high priority traffic to get through at the expense of other traffic in a device having multiple output queues and/or multiple thresholds rather than being subjected to a pure FIFO processing modality.

The output queue select (OPQSEL) value derived from the codepoint determines the queue selected for a multiple queue environment. For example, one might want to assign a relatively high priority to frames carrying voice information such as in IP telephony. This would minimize dropouts and pauses and make the conversation appear to be occurring essentially in real time without significant delays. Similarly, IP video frames might be handled in the same or a similar way. Alternatively, one might assign the high priority queue to users paying an additional fee to obtain the use of the high priority queue. Those of ordinary skill in the art will realize that many possibilities are capable of being implemented with the present invention.

Detailed Implementation

Content addressable memories ("CAMs") are well known to those of ordinary skill in the art. Such memories are typically a fixed number of bits wide and a fixed number of addresses long. For example, a CAM might be 80 bits wide by 8K (8192) addresses long. A binary CAM would include at each bit position a capability of comparing a data vector, say 80 bits long, against a programmed content of the particular address. In a binary CAM, the data vector would simply be compared binary bit for binary bit to the binary contents of the given address and a determination would be rendered as to whether a match existed or not. A ternary CAM or "TCAM" adds a capability of comparing its contents not only to a data vector having 0 or a 1 at each bit position but also to a bit position having a wild card or "don't care" condition usually represented by an "x". Thus if a TCAM entry having a data vector {0, x} representing 0 in the left bit position and "don't care" in the right bit position is compared to an input data vector having the value {0,1} there will be a match. There will also be a match if the input data vector has the value {0,0}. However, the values {1,0} and {1,1} for the input data vector would both yield a no match condition. In certain types of addressing schemes, certain bits are more meaningful than other bits, thus this ability to have a "don't care" selection (in effect, to mask certain bits) can be very useful.

A method of using a TCAM (or CAM) is to take a data vector and test it sequentially against each address of the TCAM until a match is found, then to use the address of the match to index to a location in memory containing an appropriate response to the match condition. Another method is to apply the data vector essentially simultaneously to all addresses in the TCAM or CAM and to index off of a match, if any are found. In case of multiple matches, a method of resolving the multiple match is required. Preferably, the first match is used and the rest of the entries are ignored to provide priority to the first match. A match is always guaranteed in accordance with a presently preferred embodiment of the present invention by providing a default match for instance where no other match is found.

Every frame passing through the device is preferably checked simultaneously against two tables:

(1) an L3 table implemented using a netflow switching content addressable memory (CAM); and

(2) an aggregate table using an access control list CAM preferably implemented as a ternary CAM (TCAM).

The netflow switching CAM has associated with each entry (corresponding to each active flow) a packet counter, a number of bytes seen counter, a token bucket count, and a contract value in terms of rate and bucket size. A token bucket algorithm is employed on each flow to determine whether packets are in or out of profile and/or what threshold (OPQTH) to assign. All updates to the CAM are preferably done in hardware. The default OPQTH value can be overridden for solicited bandwidth reservations (e.g., RSVP flows) only.

The Access Control List CAM (ACL CAM) preferably provides single clock cycle accesses when performing a match check for each packet. This leaves plenty of bandwidth to perform an additional QoS lookup based upon the best match for the current packet. The best match is order dependent for the entry in the ACL CAM, and may represent any field in the packet upon which the administrator wishes to base rate limitation decisions. More than one CAM entry can produce the same n-bit CAM index. The n-bit CAM index selects one of 2^N internal hardware counters, and associated preconfigured contract levels, which become affected by the packet statistics. The same or a similar token bucket algorithm applied in the netflow CAM counters is applied on these counters, allowing aggregation of traffic to be processed in parallel. The processing results from the netflow CAM and the aggregate counters are combined to produce a final new codepoint or drop decision for the current packet. Because this QoS approach is applied at the hardware level, it can run at the line rate of the system and avoid any effect on the overall switching performance of the system.

Potentially a match will be found in both tables (the L3 table and the aggregate table) based upon two independent match criteria. As pointed out above, the aggregate table will always produce a match with at least a default mask. Both tables maintain a last-seen timestamp and a token bucket. When a match occurs, the two independent bucket counts are examined to determine the frame's output queue (OPQSEL) and output queue threshold (OPQTH). If either bucket count exceeds a corresponding contract value, two independent rate limitation decisions are made. Either of these decisions may result in dropping or changing the packet. Finally the two independent rate limitation decisions are resolved to produce the final rate limitation decision for the frame.

Token bucket algorithms are well known to those of ordinary skill in the art. The basic idea is to provide a method of averaging a value which may come in spurts, such as a data transmission. In accordance with a presently preferred embodiment of the present invention, a token bucket algorithm is implemented with a counter for each table entry in the aggregate table and the L3 table. The counter is incremented for each in-profile byte associated with the flow passing through the system. A minimum byte increment may be enforced for short packets. The counter is decremented by a fixed number (the "leak rate") associated with the passage of a given amount of time. The leak rate corresponds to a contract value. This has the effect that the value stored in the counter will not grow over time as long as the leak rate exceeds or equals the data throughput rate for the flow. However, where the data throughput rate exceeds the leak rate for a significant period of time, the counter value will grow.

In a presently preferred embodiment of the present invention, the actual computation of the value of the bucket is made only when a flow hit occurs. Then the bucket count is decremented by the difference between the current time and the last seen time in time units multiplied by the leak rate (per unit time) and incremented by the number of bytes in the frame that had the flow hit.

FIGS. 4A, 4B and 4C are a block diagram of the apparatus for a quality of service policy in accordance with a presently preferred embodiment of the present invention.

Turning now to FIG. 4A, the packet enters on line **73**. At hash block **74** a hash index is obtained in a conventional manner. The hash index is used to access the Layer 3 table (L3 Table) **76** which may preferably be implemented in RAM (random access memory). Hash block **74** together with L3 table **76** form L3 CAM **78**. The packet's flow is compared to active flows existing in the L3 table **76**. If a match is found, i.e., the packet is part of an active flow, then the statistics fields corresponding to the flow and stored in L3 table **76** are accessed. If no match is found, then the L3 table **76** is updated to reflect the new flow. These statistics fields may include, for each active L3 flow, a packet counter, a number of bytes seen counter, a token bucket and a contract value. If the flow is not an active flow, i.e., there is no entry corresponding to the packet's flow in the L3 Table, then a default is preferably used. Defaults may be set by the System Administrator.

The packet is also routed from line **73** to a pair of TCAM lookup operations. The first type of TCAM lookup **80** is an aggregate table lookup which provides an index to the Aggregate Table **84** and returns a two-bit priority code on line **86** for combining the two ToS values. For example, the 2-bit priority code can indicate how to handle conflicts, e.g., "use the lowest threshold of the two ToS values", or another scheme could be used as will now be clear to those of ordinary skill in the art.

The second type of TCAM lookup **82** is an L3 Table lookup. For each frame a TCAM L3 table lookup **82** is performed and provides the contract value and token bucket counter indirectly through an index that in a preferred embodiment selects 1 of 64 choices. When hardware creates an entry in the L3 table **76**, it writes these parameters into the L3 table **76** over line **89**. Later when a frame matches the entry, there are 2 sets of parameters provided:

(1) one set of parameters provided by the L3 Table lookup **82** into the TCAM; and

(2) a second set of parameters read from the L3 table **76**.

The CAM or TCAM **88** will be logically separated into a Layer 3 Table QoS policy area and an Aggregate QoS policy area.

The data from the TCAM L3 Table lookup **82** is applied as an input to MUX **90** on line **91** as is the current data from the L3 table on line **92**.

A selection value on line **94** from the L3 Table **76** selects whether to use the parameters from the TCAM L3 Table lookup **82** or the parameters from the L3 table on line **92**.

By default, the parameters coming from the TCAM L3 Table lookup **82** are used. The system can be told with software to use the parameters stored in the L3 Table **76** instead. This approach is desirable when the parameters have been modified by software intervention. The L3 Table parameters may be initially set by software prior to flow existence or overridden by software. The L3 Table initially learns its parameters by performing TCAM L3 Table lookup **82** into the TCAM **88**.

The selected information include the contract value and is applied over line **96** to the L3 table rate limitation block **98**. A token bucket is operated as discussed above over line **100** with the L3 table **76**. The outputs of L3 table rate limitation block **98** include "L3TOS", an 8-bit representation of the calculated ToS for the packet, "L3DROP", a value indicating whether or not to drop the packet based upon it being out of profile or not, and L3ALTER, a one-bit value indicating whether or not to alter the codepoint of the packet.

The aggregate table side operates similarly. The bank of aggregate counters used for token bucket analysis is pre-configured with the codepoint and the token bucket parameters to use. The priority is not stored, allowing different policies to map to the same aggregate counter (several matches may map to the same aggregate counter index, with different priorities for resolving which ToS to use, depending upon the actual flow.

The TCAM aggregate table lookup **80** into TCAM **88** provides an index on line **101** used to access the Aggregate Table **84**. The contract value and token bucket counter are used in aggregate table rate limitation **102** to produce AGTOS, the ToS based upon the aggregate table processing branch **220** of FIG. **5**A, "AGDROP", the dropping decision based upon branch **220**, and "AGALTER", a one-bit value indicating whether or not to alter the codepoint of the packet.

The packet processing described herein is based upon the DS/ToS definition. If a valid ToS/CoS is not available, e.g., for a non-802.1q and non- IP packet, a working value is derived from other sources for internal use as discussed above. For legacy ToS definitions i.e., the present ToS definition), the precedence bits from the ToS are mapped into DS/ToS values with a conventional mapping. For frames that are not IP, the 3-bit CoS field is mapped into an 8-bit ToS field with a conventional mapping. This approach is also applied if the DS/ToS field of an incoming IP frame is assumed to be invalid for some reason.

The ToS remap takes any input ToS and maps it to a final AGTOS or L3TOS. It is configured by software. The meaning of the various possible values of the 8-bit ToS may be set by software as desired.

Turning now to FIG. **4**B, a method and apparatus for combining certain bits of L3TOS and AGTOS into a resulting one-bit TOSL3AG value in accordance with a presently preferred embodiment of the present invention is shown. A programmable compare mask **104** is used to mask bits which will not be used in the comparison. Then the two masked signals are applied to a comparing MUX **106**—providing a one-bit indication of which value is larger.

Turning now to FIG. **4**C, a method and apparatus for resolving L3TOS, L3DROP, L3ALTER, AGTOS, AGDROP and AGALTER using the two-bit priority value "priority" from FIG. **4**A is shown in accordance with a presently preferred embodiment of the present invention.

TOSL3AG, L3ALTER, AGALTER and the two-bit priority value are applied to a programmable 5:1 decoder **106**. Using a selected mechanism to resolve the various inputs (it would be as simple as "always choose L3TOS"), a bit on select line **108** to MUX **110** chooses L3TOS or AGTOS which is then provided on line **112**. Optionally certain bits of the original ToS ("ORIGTOS") may be passed through and used to override the value on line **112** using bit mask **114** and MUX **116**. The output of this process on line **118** is applied to 8:3 CoS Mapping **120** which results in a 3-bit output on line **122**. This is in turn, optionally applied to MUX **124** where, if the "DO NOT CHANGE" signal **126** is asserted, the original CoS value "ORIGCOS" on line **128** is passed as CoS_F on line **130**, otherwise the value of CoS on line **122** is passed as CoS_F on line **130**.

Similarly, the calculated ToS on line **118** is applied to MUX **132** where, if the "DO NOT CHANGE" signal **126** is asserted, the original ToS value "ORIGTOS" on line **134** is passed as ToS_F on line **136**, otherwise the value of ToS on line **118** is passed as ToS_F on line **136**.

Finally, L3DROP and AGDROP are combined and resolved as follows. The two-bit priority value, L3DROP and AGDROP are applied to a 4:1 programmable decoder **138** to obtain a dropping decision in accordance with a programmable policy. Preferably the priority value is used to select L3DROP or AGDROP. Other policies could also be programmed, such as, for example, "always use L3 DROP." The result is output on line **140**. A device such as a programmable 2:1 encoder **142** combines the signal on line **140** with a "DO NOT CHANGE" signal on line **126** to yield a signal $DROP_F$ on line **144** which follows the signal on line **140** unless "DO NOT CHANGE" is asserted, whereupon the value of the signal on line **144** is set to "DO NOT DROP."

FIGS. **5**A, **5**B, **5**C and **5**D are a flow chart detailing an implementation of a presently preferred embodiment of the present invention. At reference numeral **200** the process starts with the arrival of a packet at reference numeral **202** at a node of the communications network. For packets having a CoS field and/or a ToS field, this information is extracted at reference numeral **204**. Optionally, at reference numeral **206** it is possible to bypass some or all of the packet processing if the packet came from a "trusted source", that is, one that is already implementing a similar process in accordance with the policy implemented by the network administrator. Where the packet comes from a trusted source (as can be detected by knowing the physical port of the device that it arrived on) then a full bypass or partial bypass can be implemented. In a full bypass, as at reference numeral **208**, control is shifted to reference numeral **210** in FIG. **5**D, discussed below. In a partial bypass, as at reference numeral **212**, control is shifted to reference numeral **214** in FIG. **5**B. This is also discussed below.

If the packet is not from a trusted source, or if bypassing is not implemented, control is passed to reference numeral **216** in FIG. **5**A. At reference numeral **216** the packet is parsed for its micro flow. In this process, the pertinent part of the flow is extracted for use in accessing the CAMs associated with the Layer 3 Table and/or the Aggregate Table.

Now, in accordance with a presently preferred embodiment of the present invention, control passes in parallel along branches **218** and **220** proceeding from to reference numeral **216**. Branch **218** processes information using the Layer 3 Table approach discussed above. Branch **220** processes information using the ACL CAM/Aggregate Table approach discussed above. While it is preferred to do both in parallel, either can be used exclusively and is still within the inventive concepts disclosed herein.

Following branch **218**, the micro flow is compared to the entries in the Layer 3 Table at reference number **222**. The closest match will result in obtaining either directly, or through a pointer, the OPQSEL (output queue select) and OPQTH (output queue threshold) values for the micro flow (assuming that the micro flow has been seen recently and is therefore contained in the Layer 3 Table). In accordance with a presently preferred embodiment of the invention, the OPOSEL can be either 0 or 1 representing two output queues and the OPQTH can be **0**, **1**, **2** or **3** representing four levels of threshold. The three-bit CoS value is simply the OPQSEL bit and the two OPQTH bits. This value is sent to the port to control output queue selection and threshold. Control is then transferred at **224** to reference numeral **214** of FIG. **5**B.

Similarly, following branch **220**, the micro flow is masked at reference numeral **226** and compared to the entries in the ACLCAM/Aggregate Table. Preferably, the first match is reported and an OPQSEL and OPQTH value derived therefrom. At reference numeral **228**, control is transferred to reference numeral **214** of FIG. **5**B.

The process starting at reference numeral **214** is performed for both branch **218** and branch **220** separately.

If the policy is so set that rate limiting is in effect at reference numeral **230**, control transfers to the token bucket process starting at reference numeral **232**. Otherwise, at reference numeral **234**, control is transferred to reference numeral **236** of FIG. **5**D.

The token bucket works as follows. At reference numeral **232** a byte count denoted "BYTECOUNT 1" is read from the data store associated with the L3 table or the aggregate table. BYTECOUNT 2 is set to BYTECOUNT 1+MAX (MINBYTES, PKTBYTES), that is to say that the byte counter is set to be incremented by the larger of the number of bytes in the present packet or some minimum number of bytes which will be attributed to small packets. This is done to take into account the fact that small packets have a larger real overhead to the communications network than their respective byte counts would tend to indicate, thus they are treated as if they have an artificially larger number of bytes. This process is optional. BYTECOUNT 3 is set to the original value of BYTECOUNT 1 to hold it for future use detailed below.

Once the byte count is determined at reference numeral **232**, control transfers to reference numeral **238**. At reference numeral **238**, a determination is made as to whether the minimum time stamp interval has elapsed since the last packet was processed which matches the characteristics of the micro flow being processed. If not, reference numeral **240** is skipped. If the minimum interval has elapsed, then reference numeral **240** decrements BYTECOUNT 2 by the leak rate (LEAKRATE) multiplied by the elapsed time (# intervals). Hence that value is the leak rate per unit interval multiplied by the number of intervals elapsed based upon the last seen timestamp and current time stamp values. The last seen time stamp is preferably stored in the pertinent table with the pertinent micro flow information. After reference numeral **240**, control passes to node D, **242** and then to reference numeral **244**.

At reference numeral **244** BYTECOUNT 2 is compared to the contract value for the flow read from the appropriate data store. If BYTECOUNT 2 exceeds the contract value then the packet is out of profile and control passes to reference numeral **246** if not, the packet is in profile and control passes to reference numeral **248**.

At reference numeral **248** BYTECOUNT 1 in the data store associated with the table is updated to the value of BYTECOUNT 2. Control then passes to node C, **236** via reference numeral **250**. At reference numeral **246**, since the packet is out of profile, the BYTECOUNT 1 value in the data store associated with the table is updated for leak rate but is not charged for bytes associated with the packet. Hence, BYTECOUNT 1=BYTECOUNT 3—(LEAK RATE * # INTERVALS). Control then passes to reference numeral **252** where a value for the codepoint (ToS/CoS) is determined. This value will preferably incorporate an output threshold (OPQTH) which increases the likelihood that the packet will be dropped in various congestion situations, as it is out of profile. Control then passes to node C, **236** via reference numeral **254**.

Turning now to FIG. **5**D, from reference numeral **236**, control passes to Sao reference numeral **256** where the two ToS values (L3 ToS and AG ToS) derived from branch **218** and branch **220**, respectively, are compared and resolved as discussed above in conjunction with the discussion of FIGS. **4**B and **4**C. Control is then passed to reference numeral **258** where a drop/no drop decision is made based upon policy, only if BYTECOUNT 2 is greater than the contract value associated with the packet flow. If the decision is made to drop the packet, it is simply forwarded to no port at reference numeral **260**, otherwise control passes to reference numeral **210** and from there to reference numeral **262** where the packet is sent to a selected output port. At reference numeral **264** the final ToS is sent to the output port. At reference numeral **266** the port selects the output queue based upon the ToS/CoS. At reference numeral **268** the port optionally implements WRED on the selected queue. At reference numeral **270** if the packet is an IP packet, control may be optionally transferred to reference numeral **272** so that the packet's DS/ToS field **26** may be rewritten to incorporate the calculated ToS. At reference numeral **274** if the packet has an 802.1q tag and CoS field, control may be optionally transferred to reference numeral **276** so that the packet's 802.1q CoS field **24** may be rewritten to incorporate the calculated CoS. Optionally the CoS field may be incorporated into the packet with ISL encapsulation where it can be used downstream. The process is complete at reference numeral **278**.

Alternative Embodiments

Although illustrative presently preferred embodiments and applications of this invention are shown and described herein, many variations and modifications are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those of skill in the art after perusal of this application. The invention, therefore, is not to be limited except in the spirit of the appended claims.

What is claimed is:

1. An apparatus for implementing a quality of service policy, comprising:

a packet input;

a first flow information extractor having a single mask, said mask extracting specified flow information from packets appearing at said packet input;

a first content addressable memory (CAM) containing entries corresponding to active packet flows passing through the apparatus, said first CAM responsive to said specified flow information to provide first quality of service parameters;

a second flow information extractor having multiple masks, said multiple masks extracting aggregate flow information from packets appearing at said packet input; and

a second CAM containing entries corresponding to user-configurable aggregations of packet flows passing through the apparatus, said second CAM responsive to aggregate flow information to provide, in conjunction with an aggregate flow table, second quality of service parameters.

2. The apparatus of claim **1**, further comprising:

a first rate limiter associated with said first CAM, said first rate limiter generating at least a first codepoint for each packet appearing at said packet input; and

a second rate limiter associated with said second CAM and said aggregate flow table, said second rate limiter generating at least a second codepoint for each packet. appearing at said packet input.

3. The apparatus of claim 2, further comprising:

a codepoint resolver for applying pre-configured policy inputs to said first codepoint and said second codepoint to generate a final codepoint for each packet appearing at said packet input.

4. The apparatus of claim 3, further comprising:

a packet modifier responsive to said final codepoint for modifying CoS fields of 802.1q and ISL packets prior to transmission from the apparatus.

5. The apparatus of claim 3, further comprising:

a packet modifier responsive to said final codepoint for modifying ToS/Differentiated Service fields of IP packets prior to transmission from the apparatus.

6. The apparatus of claim 3, further comprising:

an output port having a plurality of output queues, one of said output queues being selected based upon said final codepoint.

7. The apparatus of claim 6, wherein an output queue threshold value for said selected output queue being selected based upon said final codepoint.

8. The apparatus of claim 7, further comprising:

means for dropping at least some packets of said selected output queue when said output queue threshold value is exceeded by an average queue depth at said selected output queue.

9. A method for implementing a quality of service policy, the method comprising:

receiving packets at a packet input;

extracting specified flow information from packets appearing at the packet input;

using the specified flow information, determining a match with an entry in a first content addressable memory (CAM), the first CAM containing entries corresponding to active packet flows, the first CAM responsive to the specified flow information to provide first quality of service parameters;

extracting aggregate flow information from packets appearing at the packet input; and

using the aggregate flow information, determining a match with an entry in a second CAM associated with an aggregate flow table, the second CAM containing a plurality of entries, each of the entries corresponding to user-configurable aggregations of packet flows and containing second quality of service parameters.

10. The method of claim 9, further comprising:

generating at least a first codepoint for each packet appearing at the packet input, the at least a first codepoint being associated with the first CAM; and

generating at least a second codepoint for each packet appearing at the packet input, the at least a second codepoint being associated with the second CAM.

11. The method of claim 10, further comprising:

applying pre-configured policy inputs to the at least a first codepoint and the at least a second codepoint to generate a final codepoint for each packet appearing at the packet input.

12. The method of claim 11, further comprising:

modifying CoS fields of 802.1q and ISL packets prior to transmission.

13. The method of claim 11, further comprising:

modifying ToS/Differentiated Service fields of IP packets prior to transmission.

14. The method of claim 11, further comprising:

selecting an output queue based upon the final codepoint.

15. The method of claim 14, further comprising:

based upon the final codepoint, selecting an output queue threshold value for the selected output queue.

16. The method of claim 15, further comprising:

dropping at least some packets of the selected output queue when the output queue threshold value is exceeded by an average queue depth at the selected output queue.

17. An apparatus for implementing a quality of service policy, the apparatus comprising:

a packet input for receiving packets;

means for extracting specified flow information from packets appearing at the packet input;

means, using the specified flow information, for determining a match with an entry in a first content addressable memory (CAM), the first CAM containing entries corresponding to active packet flows, the first CAM responsive to the specified flow information to provide first quality of service parameters;

means for extracting aggregate flow information from packets appearing at the packet input; and

means, using the aggregate flow information, for determining a match with an entry in a second CAM associated with an aggregate flow table, the second CAM containing a plurality of entries, each of the entries corresponding to user-configurable aggregations of packet flows and containing second quality of service parameters.

18. The apparatus of claim 17, further comprising:

means for generating at least a first codepoint for each packet appearing at the packet input, the at least a first codepoint being associated with the first CAM; and

means for generating at least a second codepoint for each packet appearing at the packet input, the at least a second codepoint being associated with the second CAM.

19. The apparatus of claim 18, further comprising:

means for applying pre-configured policy inputs to the at least a first codepoint and the at least a second codepoint to generate a final codepoint for each packet appearing at the packet input.

20. The apparatus of claim 19, further comprising:

means for modifying CoS fields of 802.1q and ISL packets prior to transmission.

21. The apparatus of claim 19, further comprising:

means for modifying ToS/Differentiated Service fields of IP packets prior to transmission.

22. The apparatus of claim 19, further comprising:

means for selecting an output queue based upon the final codepoint.

23. The apparatus of claim 22, further comprising:

means, based upon the final codepoint, for selecting an output queue threshold value for the selected output queue.

24. The apparatus of claim 23, further comprising:

means for dropping at least some packets of the selected output queue when the output queue threshold value is exceeded by an average queue depth at the selected output queue.

* * * * *

US006341130B1

(12) **United States Patent**
Lakshman et al.

(10) **Patent No.: US 6,341,130 B1**
(45) **Date of Patent: Jan. 22, 2002**

(54) **PACKET CLASSIFICATION METHOD AND APPARATUS EMPLOYING TWO FIELDS**

(75) Inventors: **Tirunell V. Lakshman**, Morganville; **Dimitrios Stiliadis**, Matawan, both of NJ (US)

(73) Assignee: **Lucent Technologies, Inc.**, Murray Hill, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/146,122**

(22) Filed: **Sep. 2, 1998**

Related U.S. Application Data

(60) Provisional application No. 60/073,996, filed on Feb. 9, 1998.

(51) **Int. Cl.**[7] **H04L 12/66**

(52) **U.S. Cl.** **370/389**; 370/351; 370/392; 370/401; 709/238; 709/245

(58) **Field of Search** 370/389, 390, 370/400, 401, 402, 403, 404, 405, 406, 407, 408, 410, 466, 467, 351, 428, 392; 709/238, 230, 246, 249, 235, 245

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,781,772 A	*	7/1998	Wilkinson, III et al.	370/401
5,951,651 A	*	9/1999	Lakshman et al.	370/389
5,995,971 A	*	11/1999	Douceur et al.	370/408
6,147,976 A	*	11/2000	Shand et al.	370/392

OTHER PUBLICATIONS

Bailey et al., PATHFINDER: A Pattern–Based Packet Classifier, University of Arizona, pp. 1–9, 1994.*

McCanne et al, The BSD Packet Filter: A New Architecture for User–level Packet Capture, Lawrence Berkley Laboratory, pp. 1–11, 1992.*

Miei et al., Parallelization of IP–Packet Filter Rules, Nippon Telegraph and Telephone Co., pp. 381–388, 1997.*

DeBerg et al., Two–and Three–Dimensional Point Location in Rectangular Subdivision, University of British Columbia, pp. 1–17, 1995.*

Alessandri, Access Control List Processing In Hardware, Diploma Thesis, ETH, pp. 1–85, 1997.*

* cited by examiner

Primary Examiner—Wellington Chin
Assistant Examiner—Frank Duong
(74) *Attorney, Agent, or Firm*—Steve Mendelsohn; Ian M. Hughes

(57) **ABSTRACT**

A packet filter for a router performs generalized packet filtering allowing range matches in two dimensions, where ranges in one dimension at least one dimension is defined as a power of two. To associate a filter rule with a received packet EP, the packet filter employs a 2-dimensional interval search and memory look-up with the filter-rule table. Values of s_m of filter-rule $r_m=(s_m,d_m)$ in one dimension are desirably ranges that are a power of two, such as prefix ranges, which are represented by a binary value having a "length" defined as the number of bits to of the prefix. The d_m may be single points, ranges defined as prefix ranges, and/or ranges defined as continuous ranges. The packet filter employs preprocessing of the filter-rules based on prefix length as a power of 2 in one dimension and decomposition of overlapping segments into non-overlapping intervals in the other dimension to form the filter-rule table. A preprocessing algorithm searches in one dimension through filter rules and arranges the corresponding filter-rule rectangle segments according to prefix length. Then, in the other dimension, the overlapping filter rectangle segments are decomposed into non-overlapping intervals, and the highest priority filter-rule overlapping each non-overlapping interval is associated with that interval. A filter-rule table is then constructed with entries ordered according to prefix length and non-overlapping interval, each entry associated with a particular filter-rule. A packet classification algorithm then matches the field or other parameter information in the packet to the filter-rule table entries to identify the filter-rule rectangle associated with the filter-rule to be applied to the packet.

30 Claims, 9 Drawing Sheets

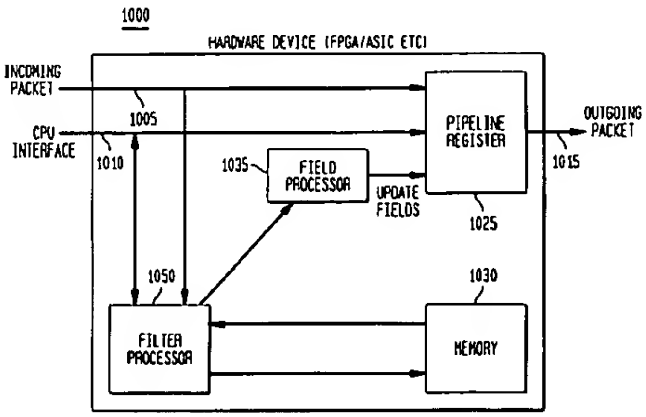


FIG. 1
(PRIOR ART)

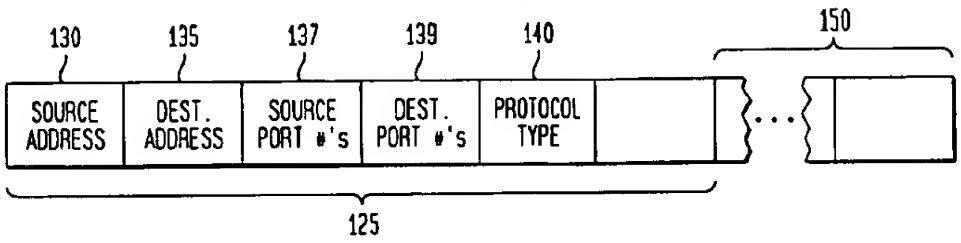


FIG. 2
(PRIOR ART)

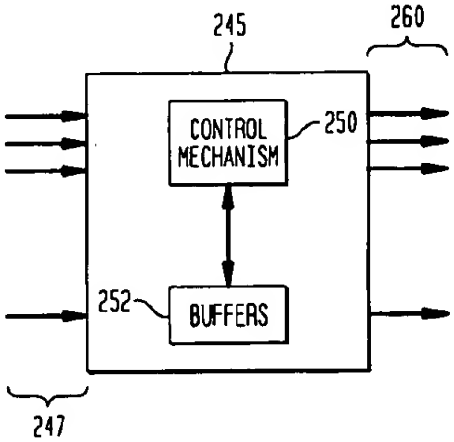


FIG. 3

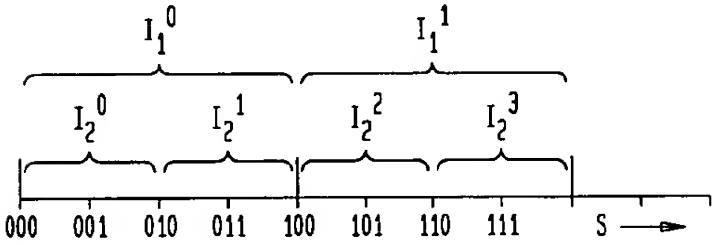


FIG. 4

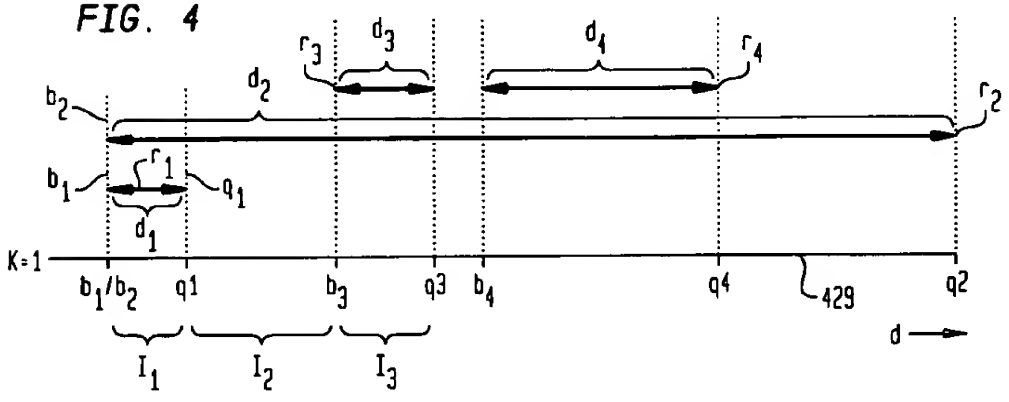


FIG. 5

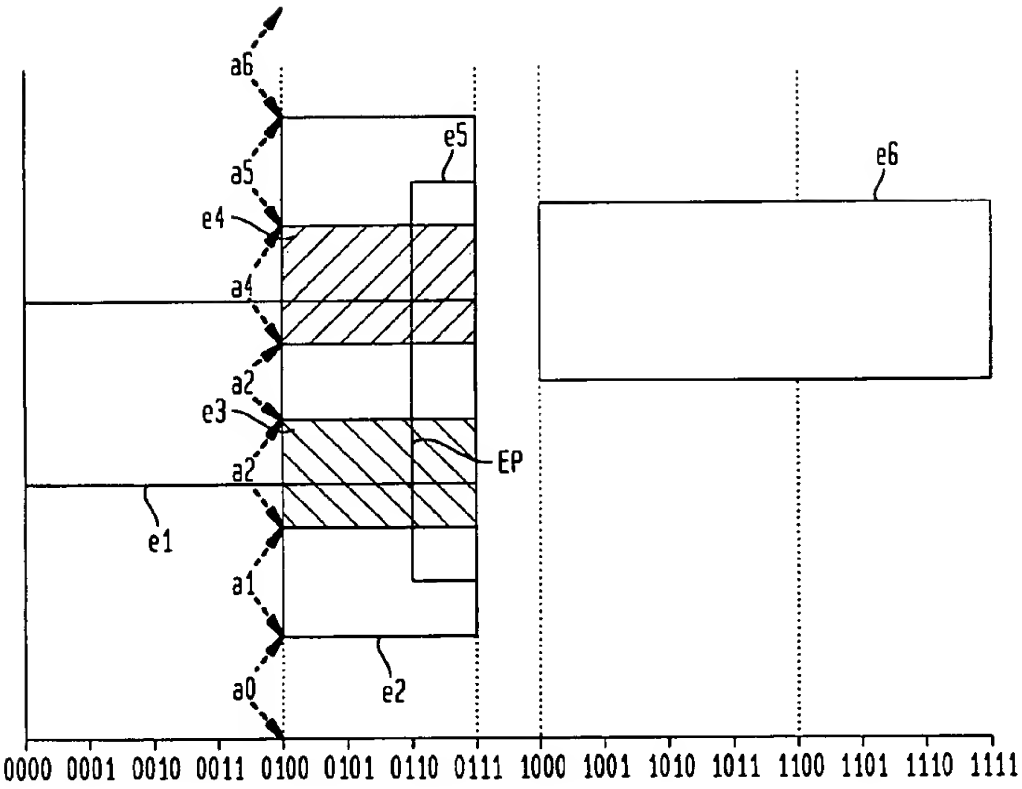

a6
a5
e4
a4
a2
e3
a2
e1
a1
a0
e2
e5
EP
e6
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

FIG. 6

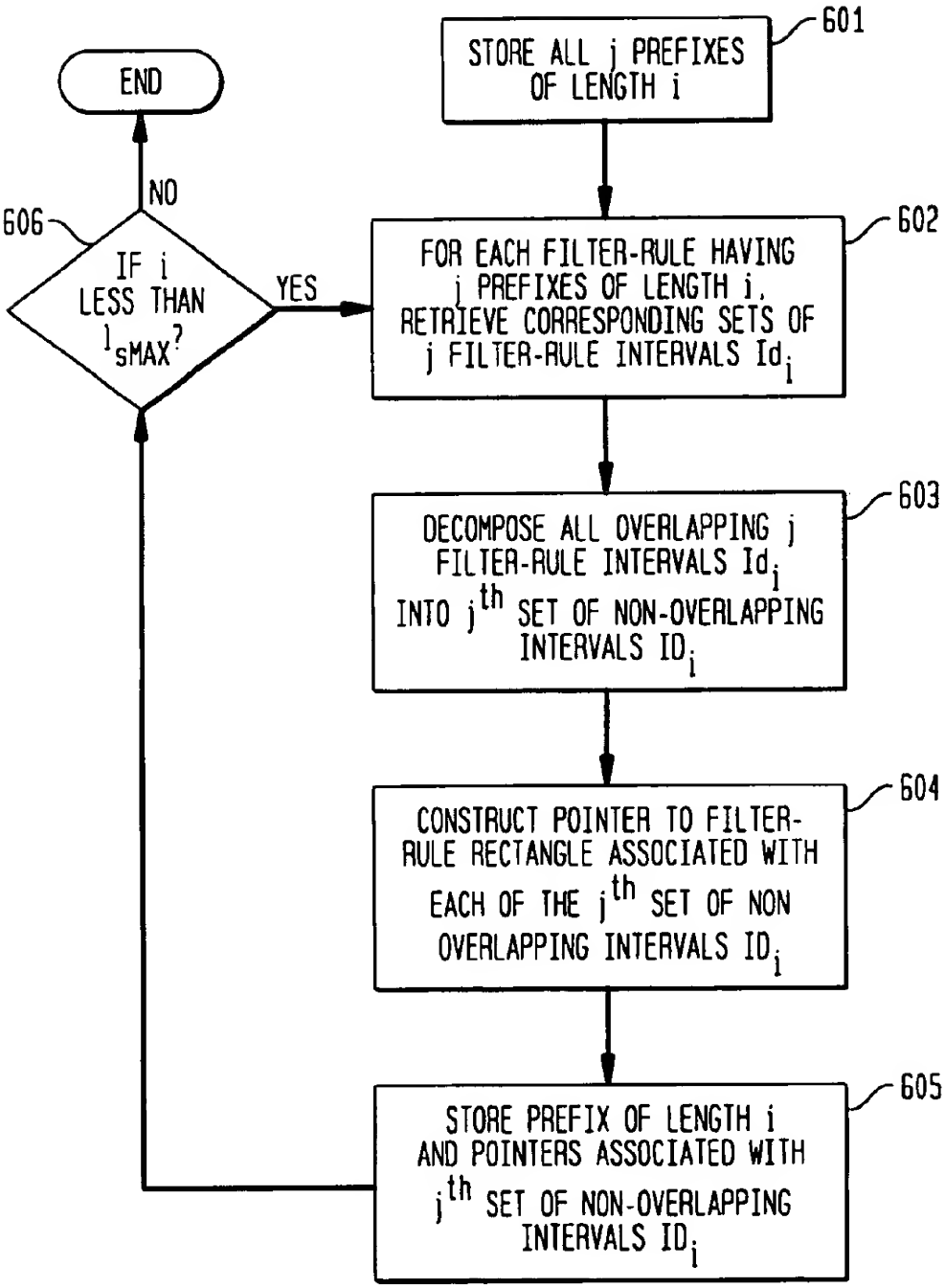

601
STORE ALL j PREFIXES OF LENGTH i
602
FOR EACH FILTER-RULE HAVING j PREFIXES OF LENGTH i, RETRIEVE CORRESPONDING SETS OF j FILTER-RULE INTERVALS Id_i
603
DECOMPOSE ALL OVERLAPPING j FILTER-RULE INTERVALS Id_i INTO j^{th} SET OF NON-OVERLAPPING INTERVALS ID_i
604
CONSTRUCT POINTER TO FILTER-RULE RECTANGLE ASSOCIATED WITH EACH OF THE j^{th} SET OF NON OVERLAPPING INTERVALS ID_i
605
STORE PREFIX OF LENGTH i AND POINTERS ASSOCIATED WITH j^{th} SET OF NON-OVERLAPPING INTERVALS ID_i
606
IF i LESS THAN l_{sMAX}?
YES
NO
END

FIG. 7

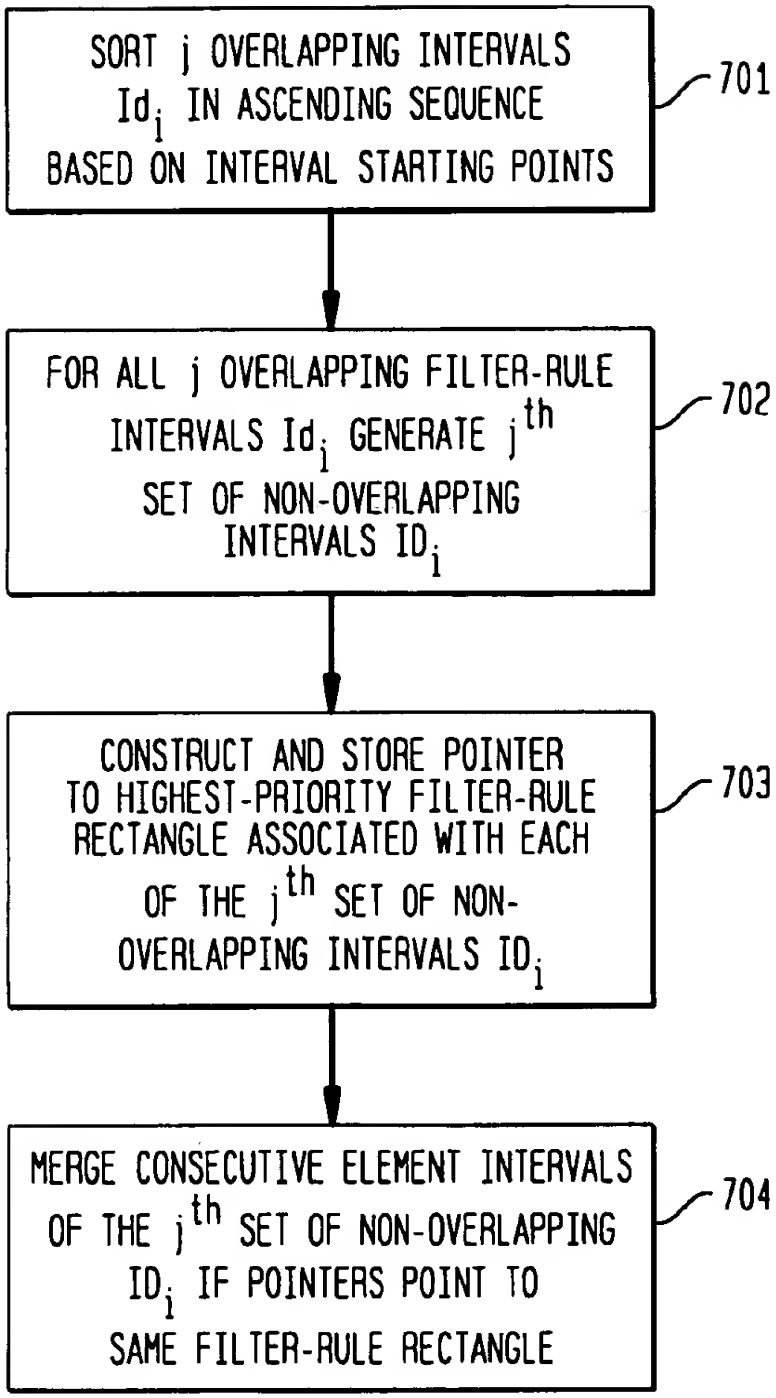
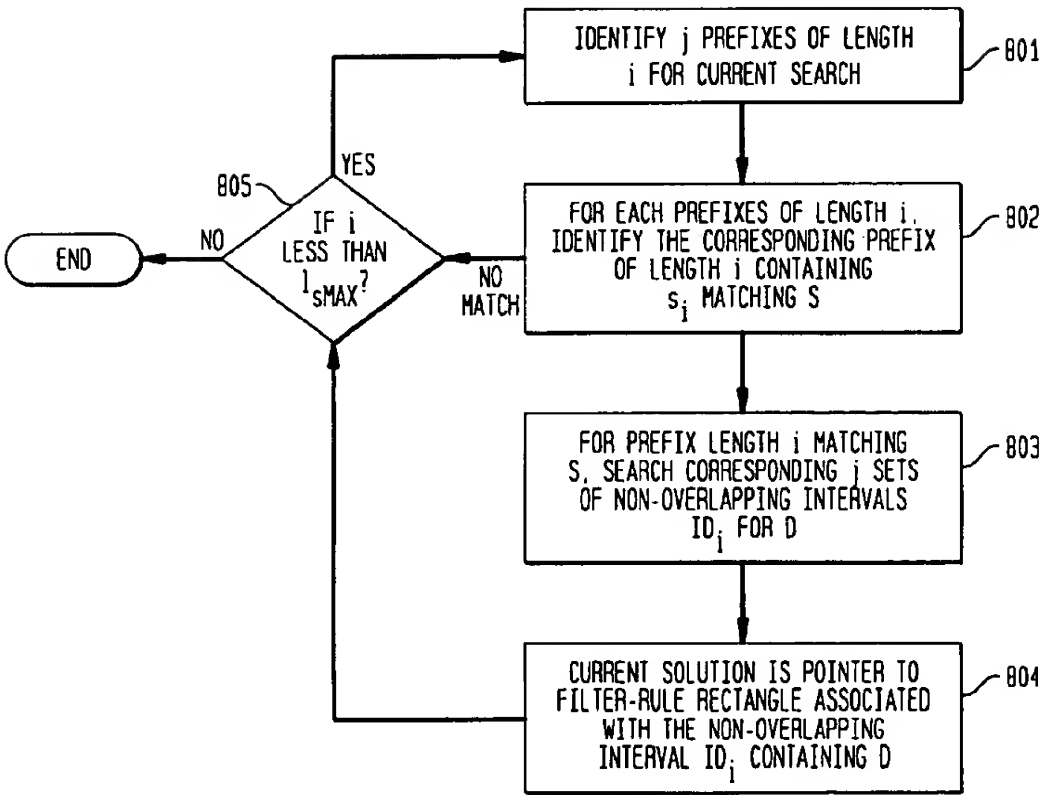

SORT j OVERLAPPING INTERVALS Id$_i$ IN ASCENDING SEQUENCE BASED ON INTERVAL STARTING POINTS
701
FOR ALL j OVERLAPPING FILTER-RULE INTERVALS Id$_i$ GENERATE jth SET OF NON-OVERLAPPING INTERVALS ID$_i$
702
CONSTRUCT AND STORE POINTER TO HIGHEST-PRIORITY FILTER-RULE RECTANGLE ASSOCIATED WITH EACH OF THE jth SET OF NON-OVERLAPPING INTERVALS ID$_i$
703
MERGE CONSECUTIVE ELEMENT INTERVALS OF THE jth SET OF NON-OVERLAPPING ID$_i$ IF POINTERS POINT TO SAME FILTER-RULE RECTANGLE
704

FIG. 8


IDENTIFY j PREFIXES OF LENGTH i FOR CURRENT SEARCH
801
FOR EACH PREFIXES OF LENGTH i, IDENTIFY THE CORRESPONDING PREFIX OF LENGTH i CONTAINING s_i MATCHING S
802
FOR PREFIX LENGTH i MATCHING S, SEARCH CORRESPONDING j SETS OF NON-OVERLAPPING INTERVALS ID_i FOR D
803
CURRENT SOLUTION IS POINTER TO FILTER-RULE RECTANGLE ASSOCIATED WITH THE NON-OVERLAPPING INTERVAL ID_i CONTAINING D
804
805
IF i LESS THAN l_{sMAX}?
YES
NO
NO MATCH
END

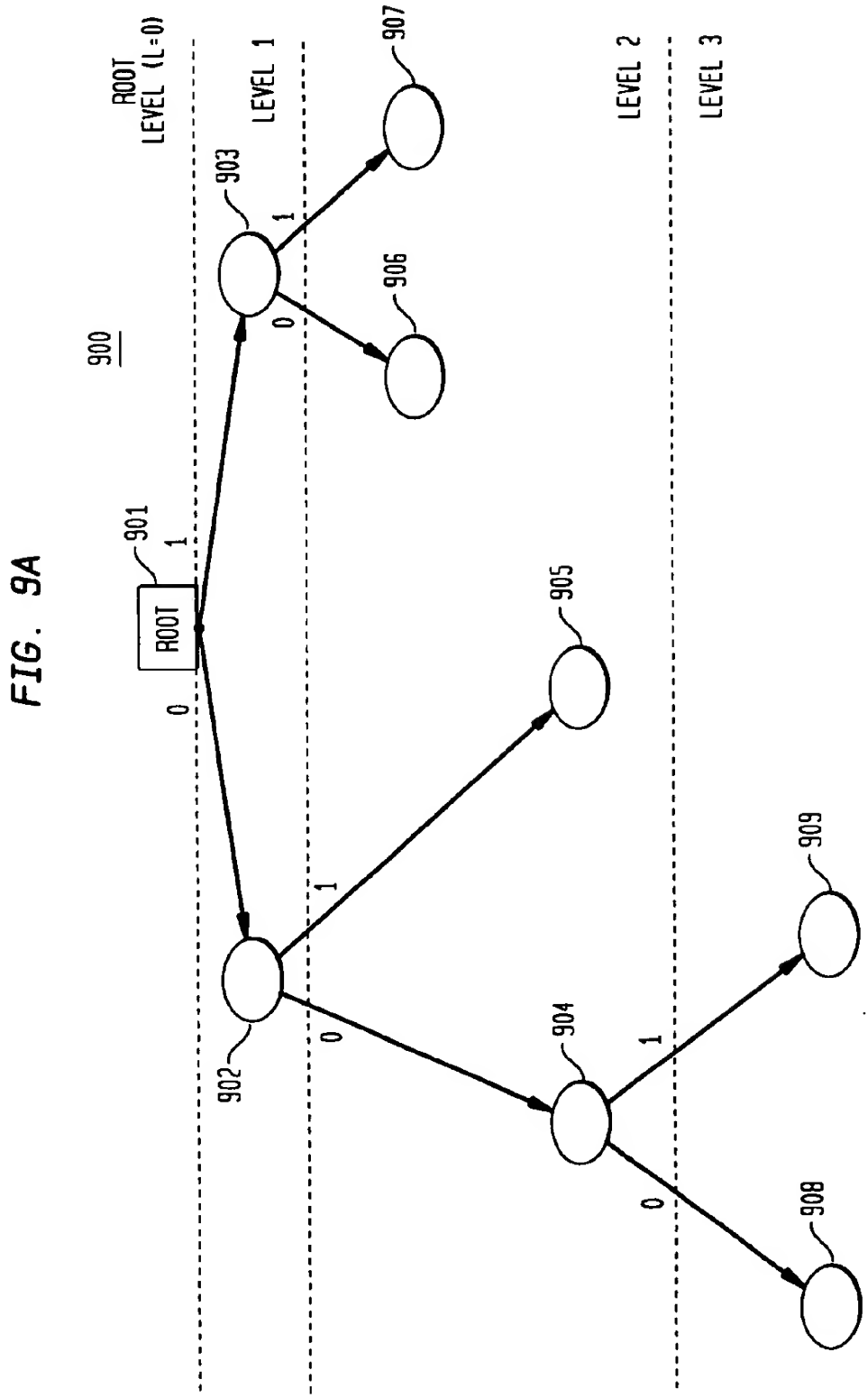

900
ROOT
LEVEL (L=0)
LEVEL 1
LEVEL 2
LEVEL 3
ROOT
901
902
903
904
905
906
907
908
909
0
1

FIG. 9A

FIG. 9B

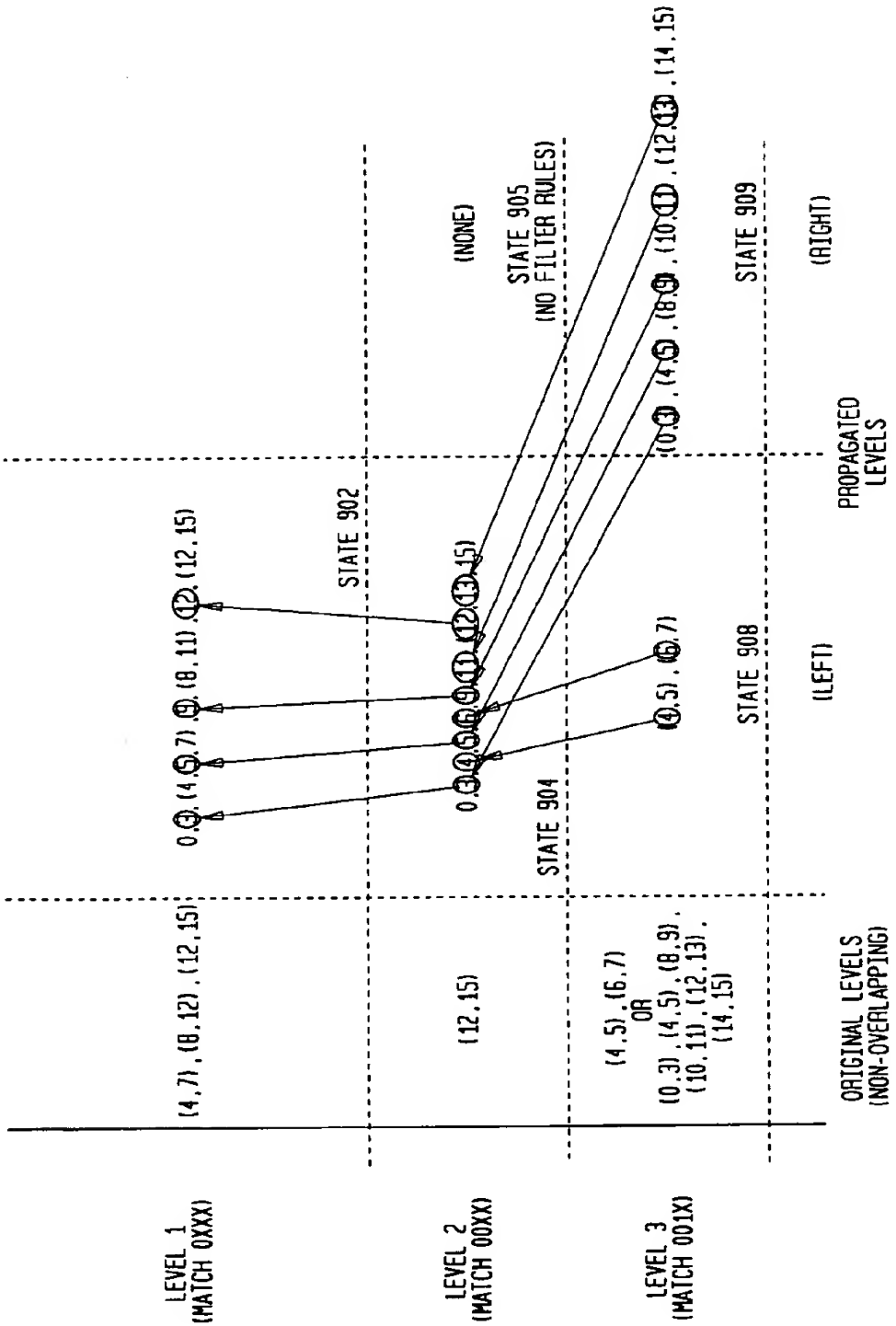

LEVEL 1
(MATCH 0XXX)
LEVEL 2
(MATCH 00XX)
LEVEL 3
(MATCH 001X)
ORIGINAL LEVELS
(NON-OVERLAPPING)
PROPAGATED
LEVELS
(LEFT)
(RIGHT)
STATE 902
STATE 904
STATE 905
(NO FILTER RULES)
STATE 908
STATE 909
(NONE)
(4,7) (8,12) (12,15)
(12,15)
(4,5) (6,7)
OR
(0,3) (4,5) (8,9)
(10,11) (12,13)
(14,15)

FIG. 10

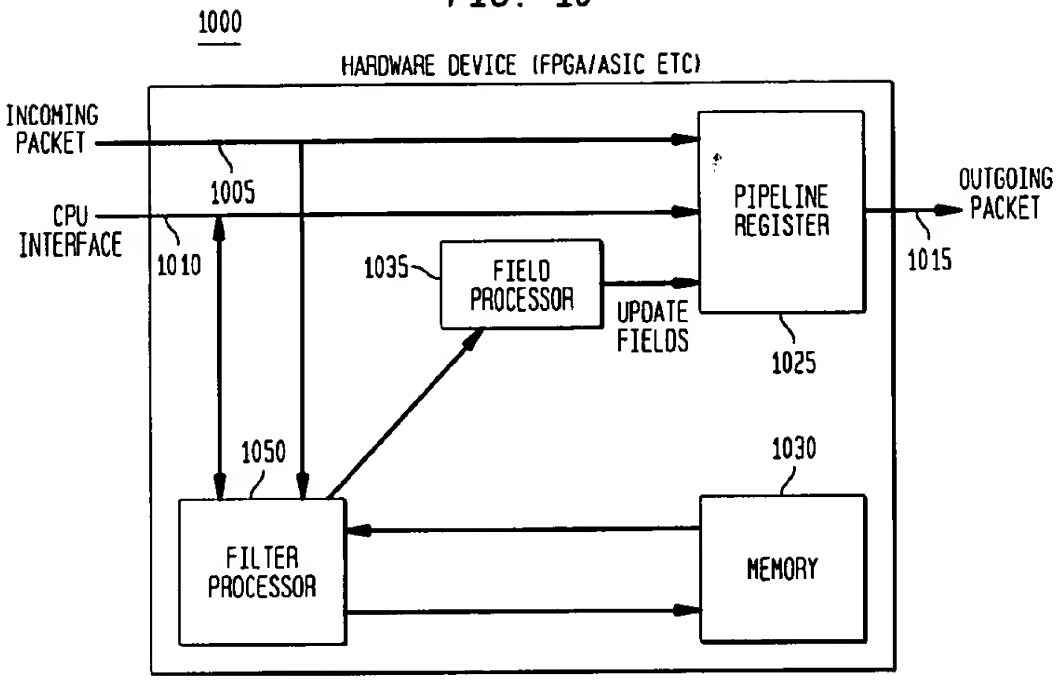

1000
HARDWARE DEVICE (FPGA/ASIC ETC)
INCOMING PACKET
1005
CPU INTERFACE
1010
1035
FIELD PROCESSOR
PIPELINE REGISTER
OUTGOING PACKET
1015
UPDATE FIELDS
1025
1050
1030
FILTER PROCESSOR
MEMORY

FIG. 11

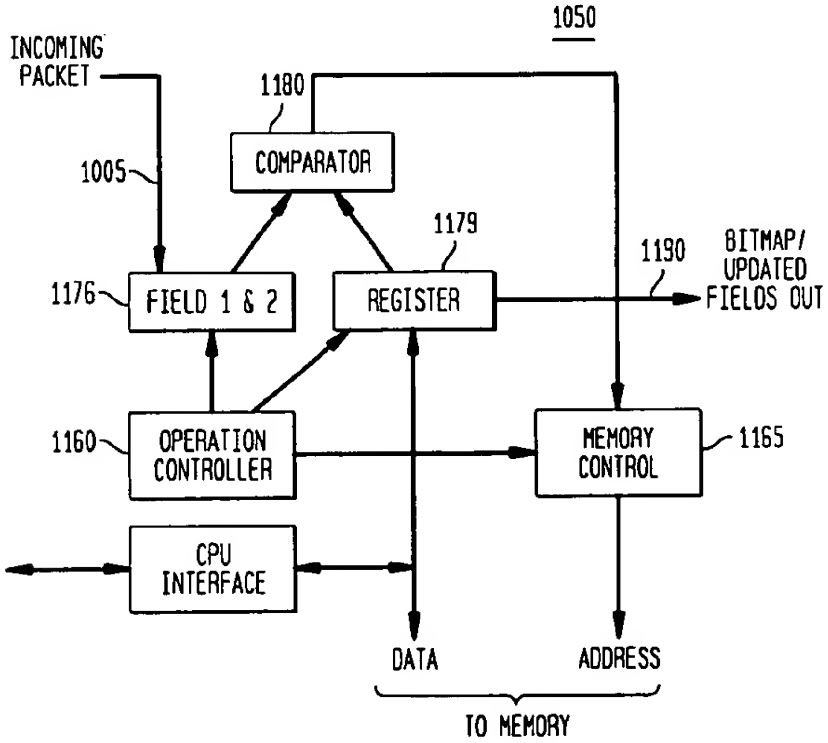
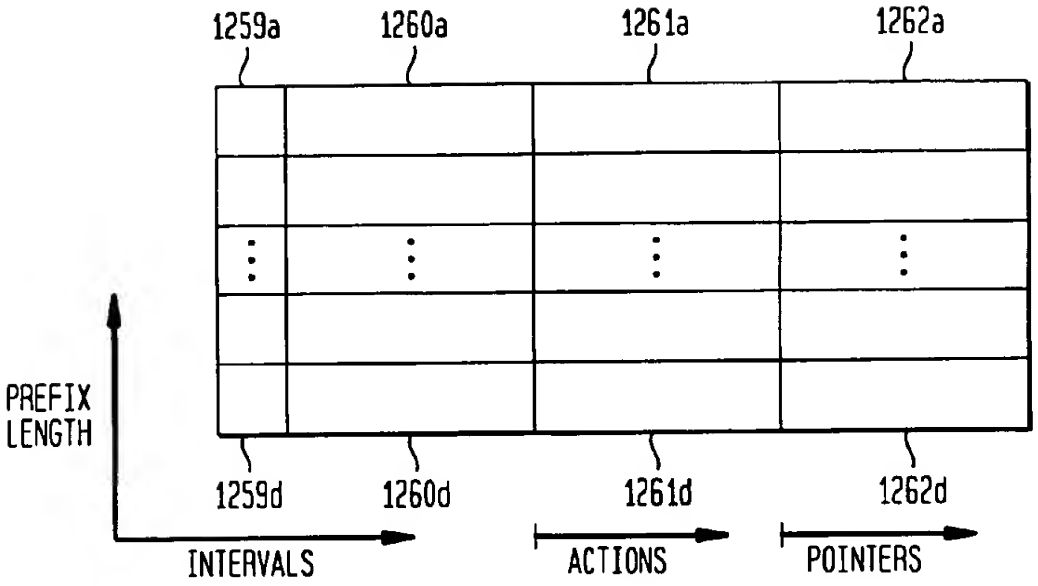

1050
INCOMING PACKET
1180
COMPARATOR
1005
1179
1190
BITMAP/ UPDATED FIELDS OUT
1176
FIELD 1 & 2
REGISTER
1160
OPERATION CONTROLLER
MEMORY CONTROL
1165
CPU INTERFACE
DATA
ADDRESS
TO MEMORY

FIG. 12


1259a
1260a
1261a
1262a
PREFIX LENGTH
1259d
1260d
1261d
1262d
INTERVALS
ACTIONS
POINTERS

PACKET CLASSIFICATION METHOD AND APPARATUS EMPLOYING TWO FIELDS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of the filing date of U.S. provisional application No. 60/073,996, filed on Feb. 9, 1998.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to packet forwarding engines used in telecommunications, and, in particular, to router algorithms and architectures for supporting packet filter operations using two packet fields.

2. Description of the Related Art

Packet-based communication networks, such as the Internet, typically employ a known protocol over paths or links through the network. Commonly known protocols are, for example, Transmission Control Protocol/Internet Protocol (TCP/IP) or Reservation Set-up Protocol (RSVP). Routers provided in a communication network provide a packet forwarding function whereby input data, usually in the form of one or more data packets, is switched or routed to a further destination along a network link. FIG. **1** shows a typical form of a data packet **20**, which may be of variable length. Data packet **20** comprises, for example, a header **125** and payload data **150**. Header **125** contains fields or parameters, such as a source address **130** where the data originates and at least one destination address **135** where the data is to be routed. Another parameter in the header **125** may be a protocol type **140** identifying a particular protocol employed in the communication network.

FIG. **2** shows a router **245** of a network node receiving streams or flows of data packets from input links **247** and routing these packet streams or flows to output links **260**. To perform a forwarding function, router **245** receives a data packet at an input link **247** and a control mechanism **250** within the router utilizes an independently generated look-up table (not shown) to determine to which output link **260** the packet should be routed. It is understood that the packet may first be queued in buffers **252** before being routed, and that the forwarding function is desirably performed at a high rate for high forwarding throughput.

Source and destination addresses may be logical addresses of end hosts (not shown). Thus, data packet **20** of FIG. **1** may further comprise unique source port numbers **137** and destination port numbers **139**. Header **125** may also include, for example, certain types of flags (not shown) in accordance with protocol type **140**, such as TCP, depending upon the receiver or transmitter application.

Network service providers, while using a shared backbone infrastructure, may provide different services to different customers based on different requirements. Such requirements may be different service pricing, security, or Quality of Service (QoS). To provide these differentiated services, routers typically include a mechanism for 1) classifying and isolating traffic, or packet flows, from different customers, 2) preventing unauthorized users from accessing specific parts of the network, and 3) providing customized performance and bandwidth in accordance with customer expectations and pricing.

Consequently, in addition to the packet forwarding function, router **245** of FIG. **2** may perform a packet filtering function. Packet filtering may be employed, for example, as "firewall protection" to prevent data or other information from being routed to certain specified destinations within the network. To perform packet filtering, the router **245** may be provided with a table or list of filter rules specifying that routing of packets sent from one or more of specified sources is denied or that specific action is to be taken for that packet having a specified source address. Such packet filtering may be employed by layer four switching applications.

Specifically, packet filtering parses fields from the packet header **125** including, for example, both the source and destination addresses. Parsing allows each incoming packet to be classified using filter rules defined by network management software, routing protocols, or real-time reservation protocols such as RSVP.

Filter rules may also specify, for example, that received packets with fields specifying that a particular destination address should or should not be forwarded through specific output links, or that some other specific action should be taken before routing such received packets. Thus, a variety of filter rules may be implemented based on packet field information. For example, such filter rules might be based on 1) source addresses; 2) destination addresses; 3) source ports; 4) destination ports; and/or 5) any combination of these fields.

Packet filtering of the prior art generally requires either an exact match operation of the fields or a match operation defined in terms of field ranges for a filter rule. Field ranges may specify, for example, ranges of source addresses, destination addresses, source/destination port numbers, and/or protocol types. Filter rules are then applied to every packet that the router receives; that is, for each packet received by the router, every filter rule is successively applied to each packet to ascertain whether that packet is to be forwarded, restricted, or re-routed according to the filter rule. However, implementation of a large number of filter rules in a router (e.g. 500 or more) is time consuming with respect to processor execution time since all filter rules must be tested. Hence, routers implementing filters having a large number of filter rules have decreased throughput, compromising a quality of service (QoS). Thus, for a router such as router **245** to maintain a relatively high level of throughput, the filtering function must be performed at very high rate.

The IP packet header fields may contain up to 128 bits of parameter information, including source and destination addresses, physical source and destination port numbers, interface number, protocol type, etc. Each of the fields or parameters in the header may be represented as being along an axis of a dimension. The general packet classification problem of a packet filter may then be modeled as a point-location in a multi-dimensional space. One or more field values of a packet define a point in the multi-dimensional space. A packet filter rule associated with a range of values of each defines an object in the multi-dimensional space.

A point-location algorithm in a multi-dimensional space with multi-dimensional objects finds the object that a particular point belongs to. In other words, given a received point $EP=\{E_1, E_2, \ldots E_D\}$ in a space having D dimensions, find one or more of a set of n D-dimensional objects including the point (n being an integer greater then 0). The general case of D>3 dimensions may be considered for the problem of packet classification. As is known in the art, the best algorithms optimized with respect to time or space have either an $O(\log^{D-1} n)$ time complexity with O(n) space or an O(log n) time complexity with $O(n^D)$ space, where O(·) mathematically represents "on the order of." Comparing

algorithms on the basis of the order of operations is particularly useful since operations may be related to memory requirements (space) and execution time (time complexity).

Though algorithms with these complexity bounds are useful in many applications, they are not currently useful for packet filtering. First, packet filtering must complete within a specified amount of time, which generally forces a value for n to be relatively small relative to asymptotic bounds, but routers typically filter packets with a number of filter rules in the range of a few thousand to tens of thousands. Consequently, even point-location algorithms with poly-logarithmic time bounds are not practical for use in a high-speed router.

For example, router **245** desirably processes n=1K filter rules of D=5 dimensions within 1 μs to sustain a 1 million-packets-per-second throughput. However, an algorithm employed with $O(\log^{D-1} n)$ complexity and O(n) space has a $\log^4 1024$ execution time and O(1024) space, which requires 10K memory accesses (look-ups) per packet. If an O(log n) time $O(n^4)$ space algorithm is employed, then the space requirement becomes prohibitively large (greater than 1000 Gigabytes).

For the special case of two dimensions, the filter rules defined for field ranges are modeled as objects in two dimensions, for example, forming rectangles in the 2-dimensional space. For a 2-dimensional space having non-overlapping rectangles, some packet filter algorithms have logarithmic complexity and near-linear space complexity. However, these algorithms do not consider the special problem related to arbitrary overlapping rectangles in the multi-dimensional space requiring a decision of which overlapping filter rules to apply to a packet. The problem may be resolved through a priority of the longest field prefix. An algorithm of the prior art where the time complexity is O(log(log N)) is based on stratified tree searches in a finite space of discrete values. Examples of these algorithms are discussed in, for example, M. De Berg, M. van Kreveld, and J. Snoeyink, Two- and Three-dimensional Point Location in Rectangular Subdivisions, Journal of Algorithms, 18:256–277, 1995. Data structures employed by this prior art algorithm require a perfect hashing operation in every level of the tree. The pre-processing complexity, without using a randomized algorithm, of calculating the perfect hash is $O(\min(hV,n^3)$, where h is the number of hash functions that must be calculated and V is the size of the space. Consequently, for a 2-dimensional space, longest-prefix lookups may result in executions requiring 2^{32} cycles, even for a relatively small number of filter rules, even if pre-processing is only required once every several seconds.

SUMMARY OF THE INVENTION

The present invention relates to a packet filter associating at least one filter rule with a packet, each filter rule and the packet characterized by values in first and second dimensions, the filter rule to be applied to the packet by a router in a communications network. In accordance with an exemplary embodiment, a filter-rule table is provided with each entry of the filter-rule table corresponding to a prefix value having a length in the first dimension and at least one interval in the second dimension. Each prefix value matching the value of the packet in the first dimension is identified, and each interval corresponding to identified prefix values containing the value of the packet in the second dimension is retrieved. A solution interval is determined as the interval associated with the prefix value associated with a predetermined metric and containing the value of the packet in the second dimension; and the filter rule corresponding to the solution interval is associated with the packet.

In accordance with another exemplary embodiment, the filter-rule table is created by first assigning each filter-rule to one or more prefix values based on the values in the first dimension; and then projecting, for each prefix value having the same length, values of each corresponding filter rule of the prefix value onto the second dimension to define at least one filter-rule segment. Each filter-rule segment is decomposed into one or more non-overlapping intervals associated with each prefix value having the same length and corresponding filter rule in the second dimension; and a pointer is generated for each non-overlapping interval identifying each filter rule contained in the non-overlapping interval. The pointer is stored as an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval.

BRIEF DESCRIPTION OF THE DRAWINGS

Other aspects, features, and advantages of the present invention will become more fully apparent from the following detailed description, the appended claims, and the accompanying drawings in which:

FIG. **1** shows a typical form of a data packet of a communications network;

FIG. **2** shows a router of a network node receiving and forwarding packet streams;

FIG. **3** illustratively depicts prefix ranges of a field in an s-dimension where the prefix ranges are a power of two;

FIG. **4** illustratively depicts segments of a filter rule having one or more field ranges of destination addresses projected as horizontal intervals;

FIG. **5** illustrates a 2-dimensional space for an exemplary packet filter in accordance with the first embodiment of the present invention;

FIG. **6** illustrate steps of an exemplary pre-processing algorithm in accordance with the present invention;

FIG. **7** illustrate steps of decomposing overlapping intervals into non-overlapping intervals as shown in FIG. **6**;

FIG. **8** illustrates steps of an exemplary classification algorithm in accordance with the present invention;

FIG. **9A** illustrates an example of trie structure of an exemplary embodiment employing virtual intervals to reduce search time of a classification algorithm;

FIG. **9B** illustrates an example of point propagation of an exemplary embodiment employing virtual intervals to reduce search time of a classification algorithm;

FIG. **10** illustrates a hardware system for implementation of the packet filter in accordance with the present invention in a packet forwarding engine or router;

FIG. **11** shows a filter processor receiving incoming packets, storing field parameters and classifying a packet in accordance with the present invention; and

FIG. **12** shows an example memory organization of a filter-rule table for the system illustrated in FIG. **10**, which depicts a filter-rule.

DETAILED DESCRIPTION

For exemplary embodiments of the present invention, a packet filter associates a 2-dimensional filter rule with an arriving packet EP having fields S and D. For a unicast forwarding packet filter, these values S and D may be source and destination address values, respectively, of the packet. For a multicast forwarding packet filter, the value S may be

the source address value of a packet and D a group identifier (ID) that identifies the multicast group that the packet may be forwarded to. The value for S may be contained in a range of binary values s, s being associated with an axis in one dimension (the s-dimension). Similarly, the value for D may be contained in a range of binary values d, d being associated with another axis in another dimension (the d-dimension). The packet filter includes a set of n packet-filtering rules RP having 2 dimensional filter rules r_1 through r_n to be associated with the packet. Each filter rule r_m, m an integer greater than 0, may be denoted as $r_m=\{s_m,d_m\}$, which is a set of two field ranges s_m and d_m in the s-dimension and d-dimension that define the filter rule r_m in the 2-dimensional space.

To associate a filter rule with a received packet EP, the packet filter employs a 2-dimensional interval search and memory look-up with the filter-rule table. Locating a pair of values S and D for fields of a packet EP and associating a 2-dimensional filter rule with the packet may be modeled as a point-location problem in a 2-dimensional space. The packet EP having field values S and D arrives at the router and is defined as a query point (S, D) of a 2-dimensional space. For the point-location problem where packet filtering involves orthogonal rectangular ranges, a search in 2-dimensions of a 2-dimensional, orthogonal, rectangular range decomposes each rectangle into a set of 1-dimensional filter-rule intervals to allow 1-dimensional searches over 1-dimensional intervals.

For a simple embodiment, preprocessing of filter-rules may construct the filter-rule table as a 2-dimensional look-up table comprising filter-rule pairs (s_m,d_m), m an integer greater than 0, where each s_m is a prefix of possible source addresses and each d_m is a contiguous range, or a single point, of possible destination addresses or group IDs. For the table, each pair (s_m,d_m) defines a filter-rule rectangle $r_m=\{s_m, d_m\}$ for the n packet-filtering rules r_1 through r_n in 2-dimensions, and rectangles may overlap. The point location in a 2-dimensional space operates as follows: given the query point (S, D) of packet EP, the search or look-up algorithm for packet classification finds an enclosing filter-rule rectangle $r_m=(s_m,d_m)$, if any, such that the query point (S, D) is contained in r_m, and such that s_m is the most specific filter according to a predefined metric, such as, for example, the longest matching prefix of field value S or the highest priority rule for a given prefix length.

For Internet Protocol (IP) routers employing an algorithm in accordance with the present invention, look-up tables may have as many as 2^{16} entries or more. Also, algorithms employed may generally be evaluated based on worst-case performance since queuing for header processing is desirably avoided to provide a specific Quality of Service (QoS). For the exemplary filter-rule table, a value n may be defined to denote a number of entries in the table, for example a multicast forwarding table, corresponding to the n filter rules r_1 through r_n. An nxn array may be formed in a memory with each entry representing the highest-priority filter-rule rectangle of the n filter rules r_1 through r_n enclosing a point corresponding to the coordinates represented by the entry. An exemplary classification (i.e., look-up) algorithm that employs this simple table may employ two binary searches, one for each of the dimension. This exemplary classification algorithm may require O(log n) time and $O(n^2)$ memory space. The $O(n^2)$ memory space is due to one rectangle being represented in O(n) locations. Such simple table might not be preferred, however, for a high-speed router when the number of filtering rules is $n=2^{16}$ or greater since the required memory space or memory access time may be excessive.

Consequently, preferred embodiments of the present invention employ preprocessing of the filter-rules based on prefix length as a power of 2 in one dimension and decomposition of overlapping segments into non-overlapping intervals in the other dimension to form the filter-rule table. A packet filter of the present invention first searches in one dimension through filter rules and arranges the corresponding filter-rule rectangle segments according to prefix length. Then, in the other dimension, the overlapping filter rectangle segments are decomposed into non-overlapping intervals, and the highest priority filter-rule overlapping each non-overlapping interval is associated with that interval. A filter-rule table is then constructed with entries ordered according to prefix length and non-overlapping interval, each entry associated with a particular filter-rule. This filter-rule table is constructed within a router prior to processing of received packets. Packet classification in accordance with the present invention then processes the received packets using the field or other parameter information in the packet. The field or other parameter information is matched to the filter-rule table entries to identify the filter-rule rectangle associated with the filter-rule to be applied to the packet.

In accordance with the present invention, values for each s_m of $r_m=(s_m,d_m)$ in the s-dimension are desirably ranges that are a power of two. Consequently, prefix values ("prefixes") define ranges ("prefix ranges") that are a power of two. The length of a prefix is the number of specified bits of the prefix. The prefix range is between a lower bound defined by the prefix and unspecified bits set to logic "0" and the upper bound defined by the prefix and unspecified bits set to logic "1". The length may be represented by a binary value. The d_m may be single points, ranges defined in a manner similar to prefix ranges in the s-dimension, and/or ranges defined as continuous ranges. When multiple matches of a same length prefix occur for a specific value of s_m, the query point (S, D) is associated with the highest priority filter rule having the matching prefix of d_m, if an overlap also occurs in the d-dimension.

FIG. 3 illustratively depicts prefixes and prefix ranges of a field in a s-dimension where the prefix ranges are a power of two. Field values s, which may be source addresses, vary from 000 to 111 (binary). An address may be a point (i.e., 010) or within a range (i.e., 010 to 101). For a special case, prefix ranges may be a power of 2. For example, if a prefix range is defined as Oxx, the prefix, represented as a single value 0, specifies the range 000to 011. For this example, the prefix has a length of 1 corresponding to one specified bit. Two prefixes of length 1 are possible: I_1^0 and I_1^1. If the prefix has two bits, or a length of 2, then four prefixes are possible: I_2^0, I_2^1, I_2^2, and I_2^3. Prefixes of different length define prefix ranges that are different powers of two. The prefix ranges do not overlap.

FIG. 4 illustrates an example of decomposition in the d-dimension of a 2-dimensional filter-rule rectangle into 1-dimensional overlapping segment sets and then into non-overlapping intervals. As described previously, values for each d_m of filter rule $r_m=(s_m,d_m)$ in the d-dimension may be any contiguous range and are not necessarily restricted to prefix ranges only. FIG. 4 shows a horizontal axis 429 for the d-dimension representing, for example, parameter values for IP destination addresses. The process searches through each of the applicable filter rules $r_1, \ldots r_4$ to be implemented in the router for each dimension, and the process may be implemented before processing of arriving packets. Each of the filter rules $r_1, \ldots r_4$ specifies field ranges such as $d_1, \ldots d_4$ for the d-dimension applicable to the particular parameter of the packet header.

Field ranges $d_1, \ldots d_4$ are projected as overlapping horizontal line segments, with each segment specifying a start point "b_i" and end point "q_i" of a range for a particular corresponding filter rule (i an integer greater than 0). For example, d_1 specifies a first range of source addresses on a first segment defined by start point "b_1" and end point "q_1" for filter rule r_1. Segments may overlap, such as those of d_1 and d_2. Consequently, segments are decomposed into non-overlapping intervals I_j (j an integer greater than 0). Therefore, the segment defined by start point "b_1" and end point "q_1" for filter rule r_1 has a single associated interval I_1, but the segment defined by start point "b_2" and end point "q_2" for filter rule r_2 has three intervals I_1, I_2, and I_3 associated with filter rule r_2. These three non-overlapping intervals I_1, I_2, and I_3 are a result of decomposing the overlapped segments of filter rules r_1, r_2, and r_3 at start or end points. It should be understood that for each filter rule, a range of source addresses and a range of destination addresses, for example, may be specified.

As described previously, values in the s-dimension of each rectangle desirably have lengths of a power of 2 when the values in the s-dimension are defined as prefix ranges. Ranges in dimensions being prefix ranges provide constraints such as illustrated in FIG. 3. When prefix range intervals have lengths which are powers of two, arbitrary overlapping of filter-rules for the dimension does not occur since two prefixes of the same length do not overlap. Also, a prefix range interval starts from an even-value point and terminates at an odd-value point. Consequently, a set of prefix ranges form several distinct cells distinguished by the length of the prefix or, equivalently, the length of the range. Further, values for each d_m of filter rule $r_m=(s_m,d_m)$ in the d-dimension may be any contiguous range, such as illustrated in FIG. 4, and are not necessarily restricted to prefix ranges unless the value for d_m is defined as a prefix range. However, modifying the packet filter in accordance with the present invention to define values for d_m as prefix ranges may be desirable, such as if destination addresses are concatenated with layer-4 destination ports or some other similar header field.

In accordance with the present invention, filter-rule table cells for prefix ranges and associated non-overlapping intervals are defined containing pointers to filter-rules as entries in the filter-rule table in the following manner. Given each rule $r_i=(s_i,d_i)$, for the field range s_i that is an integer power of 2, the length is defined as l_{si} bits and for the field range d_i the length is defined as l_{di} bits. The maximum values of lengths l_{si} and l_{di} are defined as l_{sMAX} and l_{dMAX}, respectively. The set of prefixes having a length of i bits are denoted as P_i, $i<0 \leqq l_{sMAX}$. As described with respect to FIG. 3, there may be several different prefixes of a given length i, i.e. the set of prefixes of length 1 (P_1) may have up to two elements, prefixes starting with "0" and prefixes starting with "1". The value np_i denotes the number of elements in the set of prefixes of length i (P_i) that are present in the lookup table. The elements of the set of prefixes of length i (P_i) may be numbered in ascending order of their values; consequently, the np_i prefixes of the set P_i are defined as the set $\{P_i^1, P_i^2, \ldots, P_i^{npi}\}$.

The set of filter-rule rectangles $RP=\{RP_1, RP_2, \ldots, RP_{1sMAX}\}$ is defined such that each RP_i is a subset of the set of n filter rule rectangles RP such that subset RP_i includes all filter-rule rectangles formed from s value prefixes having a length of i bits. Further, each subset RP_i may be defined as the union of the sets of filter-rule rectangles $RP_i^j=\{(P_i^j,d_i^1), (P_i^j,d_i^2), \ldots,\}$ where each filter-rule rectangle RP_i^j has the j^{th} prefix of length i (P_i^j) as a side of the filter-rule rectangle in the s-dimension. Therefore, each of the filter-rule rectangles in set RP_i^j may associated with each prefix P_i^j (j an integer and $1 \leqq j \leqq np_i$).

Each value d_i^j in the d-dimension of the set of filter-rule rectangles $RP_i^j=\{(P_i,d_i^1), (P_i^j,d_i^2), \ldots, \}$ is a range in the d-dimension that may overlap other ranges. As defined, the subset of rectangles RP_i is the union of sets $RP_i^1, RP_i^2, \ldots, RP_i^j \ldots, RP_i^{npi}$, (j an integer and $1 \leqq j \leqq np_i$), and each of the RP_i^j are disjoint. Filter-rule rectangles in set RP_i^j are formed with longer prefixes than those filter rectangles in set RP_i^t if $i>t$. A filter-rule having a longer prefix value in the s-dimension may be defined to have higher priority than other filter-rules with shorter prefix length since they are more specific with respect to, for example, packet source address. Consequently, if filter-rule rectangles in RP_i^j and RP_i^t match a point EP=(S, D) based on field values in the s-dimension, then the filter-rule associated with RP_i^j is applied to packet EP. The filter-rule associated with RP_i^t is applied to packet EP since rectangles in RP_i^j are formed with longer prefixes than those rectangles formed in RP_i^t.

For the d-dimension, the size of the list of the set of d_i^j values may be defined as k_i^j, k an integer greater than 1. From each list of j ranges in a rule set RP_i comprising (s_i, d_i^j), a list of non-overlapping intervals ID_i^j is formed along the axis of the d-dimension from filter-rule segments Id_i^j corresponding to the values of d_i^j. The size of this new set of intervals ID_i^j may be $K_i^j \leqq 2k_i^j+1$. By representing the original k_i^j overlapping intervals as non-overlapping intervals, a memory space requirement of the packet filter may be increased by only a constant factor of 2.

For the d-dimension, if the values for d_i^j are defined to be prefix ranges, then the projected filter-rule segments Id_i^j along the d-dimension axis do not overlap, and so the Id_i^j become the list of non-overlapping intervals ID_i^j.

For the general case, replacing overlapping intervals by non-overlapping intervals allows a search algorithm to locate the field value D from the query point (S, D) on one of these non-overlapping rectangles during the search procedure. The search algorithm then retrieves the associated enclosing rectangle of the non-overlapping rectangles representing the filter rule to be applied to the packet. Consequently, when many filter-rule rectangles overlap a given interval in the d-dimension, the particular filter-rule rectangle associated with the given interval when non-overlapping intervals are formed is the filter-rule rectangle with the highest priority that overlaps the interval.

FIG. 5 illustrates a 2-dimensional space for an exemplary packet filter in accordance with the first embodiment. FIG. 5 shows a total of $np_1=2$ prefixes of length i equal to 1 (i.e. 0xxx and 1xxx). For the set of rectangles RP_1 with prefix length i equal to 1, the corresponding set of filter-rule rectangles is $RP_1=\{e1,.e6\}$. Also shown is a total of $np_2=1$ prefixes of length i equal to 2 (i.e., 01xx) for the set RP_2 of filter-rule rectangles formed with prefixes of length i equal to 2. The set RP_2 includes the filter-rule rectangles {e2, e3, e4}. These filter-rule rectangles may overlap on the axis of the d-dimension. Similarly, set of filter-rule rectangles RP_3 with prefix of length i equal to 3 (i.e., 011x) contains one filter-rule rectangle e5.

For the illustration shown in FIG. 5, the set of intervals given a prefix length of 2 that are created after this overlap elimination for each Id_2^1 is $ID_2^1=\{a_O, a_1, \ldots a_6\}$. Filter-rule rectangles e2 and e3 overlap in the d-dimension. Filter-rule rectangle e3 of the set of rectangles RP_2^1 is associated with interval a_2, since this filter-rule rectangle may be defined to have the higher priority than filter rule rectangle e2.

Consequently, only this filter-rule rectangle e3 is associated with interval a_2 even though another filter-rule rectangle with lower priority overlaps this range a_2.

For the exemplary system of FIG. 5, a packet EP with header field values (S=0110, D=0101) arrives. First, a matching prefix of length 1 from S=(0) is found and a search performed for enclosing rectangles formed with this prefix. The d-dimension is searched and filter-rule rectangle el shown in FIG. 4 is a first candidate rule, or is the current solution. Note that rectangles e1 and e6 of FIG. 5 are the only rectangles in the set of rectangles with prefixes of length equal to 1. Next, a search for the matching prefix (01) is performed over the prefixes of length 2. Rectangle e3 is determined to be a better candidate rule since 1) the D value of the arriving packet overlaps with the range a2, 2) this filter-rule rectangle e3 is formed with a longer prefix than rule el, and 3) this filter-rule rectangle has higher priority than other rectangles formed with prefixes of equal or lower length. Finally, a matching prefix (001) of length 3 is located and a search among rectangles with this prefix is performed, resulting in the rule of rectangle e5 as the best solution.

A packet filter of the present invention for a router employs an algorithm having two parts. The first part is a pre-processing algorithm that searches through filter rules and decomposes the filter rules for each dimension. The first part is performed by the router prior to processing of received packets. A second part is a classification algorithm that processes the received packets using the field or other parameter information in accordance with the processed filter rules of the pre-processing algorithm.

An exemplary pre-processing algorithm for a packet filter in accordance with the present invention is shown and described with respect to FIG. 6 and FIG. 7. The pre-processing algorithm performs three operations to decompose the n filter-rule rectangles. First, the filter-rule rectangles are separated based on the prefix length in the s-dimension. Second, for each prefix of length i, all associated filter-rule rectangles are projected onto the corresponding axis in the d-dimension to obtain first the overlapping intervals Id_i^j. Third, a set of non-overlapping intervals ID_i^j are created from these the overlapping intervals Id_i^j. The non-overlapping intervals may be created by a scan of the overlapping intervals from lower to higher coordinates in the d dimension.

FIG. 6 illustrates a flowchart of an exemplary pre-processing algorithm in accordance with the present invention. First, at step **601** the set of prefixes P_i^j (as defined previously) for all i and j, $1 \leqq i \leqq l_{sMAX}$ and $1 \leqq j1 \leqq np_i$, is stored in memory according to, for example, an efficient trie representation. Then, at step **602** for each filter-rule having prefix P_i^j, the corresponding set of filter-rule values d_i^j in the d-dimension are projected as overlapping segments Id_i^j. At step **603**, for all P_i^j, (i.e., for all j prefixes of length i, $1 \leqq i \leqq l_{sMAX}$ and $1 \leqq j1 \leqq np_i$), the overlapping segments Id_i^j are decomposed into a set of non-overlapping intervals ID_i^j. At step **604** a pointer is constructed to identify the highest priority filter-rule rectangle overlapping the associated non-overlapping interval for all intervals of the set ID_i^j. At step **605**, the set of non-overlapping intervals ID_i^j are stored with associated prefix P_i^j as table entry in the filter-rule table. Each entry of the filter-rule table corresponds to the pointer identifying actions to applied to a packet for a corresponding filter rule. The list of non-overlapping intervals ID_i^j may be stored in sorted sequence using either an array or a binary tree. At step **606**, the algorithm returns to step **602** if $i<l_{sMAX}$, or until all prefix lengths P_i are processed.

FIG. 7 is a flowchart illustrating the decomposition of intervals of the steps **603** and **604** of FIG. 6. For step **603** of FIG. 6, first, at step **701** the overlapping intervals Id_i^j are sorted into an ascending sequence based on interval starting points. Then, at step **702**, for all j, if an overlapping interval Id_i^j starts or ends, an assigned, non-overlapping interval ID_i^j is generated for previous interval. For step **604** of FIG. 6, at step **703**, the assigned, non-overlapping intervals ID_i^j and corresponding pointer to actions for the highest-priority filter-rule rectangle overlapping this interval are stored in memory. Optionally, at step **704** the newly created interval and the previously stored adjacent interval are compared, and are merged if the two intervals point to the same filter-rule. Since a new interval ID_i^j is created, at most, when an overlapping interval begins or terminates, the size of this new set of intervals ID_i^j is $K_i^j \leqq 2k_i^j+1$ where k_i^j is the size of the set of overlapping intervals Id_i^j.

In accordance with the pre-processing algorithm of the packet filter, each filter-rule is associated with a pointer in one or more filter-rule table entries. Each filter-rule pointer is stored in exactly one address in memory corresponding to prefix and prefix length on the s-dimension axis, and one or more addresses corresponding to non-overlapping intervals on the d-dimension axis. The set of filter-rule rectangles associated with a prefix is stored as a list of non-overlapping intervals and requires space only proportional to the size of the set. Only O(n) memory space may be utilized to store all the rectangles since each rectangle appears only in one set and therefore the size of the union of all sets is O(n).

Once the preprocessing algorithm creates the filter-rule table, the classification algorithm performs a look-up search of the filter-rule table. FIG. 8 illustrates an exemplary flow-chart of the classification algorithm of the packet filter. The classification algorithm may begin at step **801**. First, at step **801**, prefixes of length i, $P_i=\{P_i^1, P_i^2, \ldots, P_i^{np_i}\}$ are identified. Initially, the value of i may start from the shortest prefix length, such as i=1. Next, at step **802** the prefix P_i^j of length i with an s_i matching the query point S in the s-dimension is determined. If no match of S with s_i in P_i^j is found at step **802**, then the algorithm moves to step **805**. At step **805**, the prefix length value i is incremented, until the longest prefix length is searched (i.e. increment i if $i<l_{sMAX}$). Consequently, the classification algorithm repeats for each prefix length until all prefix lengths have been searched.

If a match of S with an s_i in P_i^j is found at step **802**, then at step **803** the stored structure in the d-dimension associated with P_i^j is searched to find the non-overlapping interval ID_i^m m that contains the query point D in the d-dimension. At step **804** the current solution is set as the pointer associated with table entry (P_i^j, ID_i^m) (m an integer greater than 0). The current solution may be the "best" solution among all prefix lengths searched so far if shorter prefix lengths correspond to lower priority rules, and the search begins at the shortest prefix (lowest priority) and goes to the longest prefix (highest priority). The algorithm then moves to step **805**.

The number of iterations of the classification algorithm in the worst case is equal to the largest number of possible prefix lengths, which is l_{sMAX}. Consequently, the total time for searching through all prefix lengths is $O(l_{sMAX})$ times the time to search a list for a prefix length. In addition, the size of the lists of ID_i^j for a prefix length may be O(n) since there are n filter-rules. Hence, an average O(log n) time is needed to search each list for a matching entry. The worst case total execution time of the exemplary classification algorithm is, therefore, $O(l_{sMAX} \log n)$.

However, for large numbers of table entries, worst case performance may not be sufficient for available processor speed. For example, if a number of possible prefix lengths

l_{sMAX} is 32 and the number of table entries n is 2^{18}=256K. This exemplary classification algorithm may perform 576 memory accesses in the worst case, which may be prohibitively high. An alternative embodiment of the present invention employs a trie structure with virtual intervals for storage of data in memory to reduce the worst-case time-complexity $O(l_{sMAX}\log n)$ to a time-complexity $O(l_{sMAX})$.

A trie structure may be employed for data storage with a memory space requirement that may be O(n). Furthermore, the order of search for the sets of filter-rules RP_1, RP_2, . . . , may be organized by increasing order of prefix lengths. For example, a set of intervals from RP_1 is searched before searching a set of intervals from RP_2 and so on. The search proceeds in levels L_i, with a search of sets belonging to RP_1 being on the first level L_i, those in RP_2 being on the second level $L_2$and so on. The number of non-overlapping intervals in all of RP_i is defined as N_i. The root (i.e, bottom-most) level R_{ie} has N_{ie} non-overlapping intervals, and this level may be RP_1 with N_1 non-overlapping intervals. The number of overlapping intervals at each level without introducing virtual intervals may be O(n). In accordance with the present invention, introducing "virtual" intervals decreases search time of the classification algorithm in multiple ordered lists. If elements of a set of intervals are arranged by employing virtual intervals as described below, the worst case execution time may be $O(l_{sMAX}+\log n)$.

A search of the list of non-overlapping intervals at level L_i, for example, yields a result of the point D, where D is in an interval ID_i^j. A search of the lists at the next level L_{i+1} is performed, instead of searching through the remaining intervals at level L_i. In general, the result of the previous search at level L_i may be used for the search at level L_{i+1}, and the search at level L_{i+1} is performed for only those intervals that fall in the range of intervals ID_{i+1}^j in level L_{i+1} given by the interval ID_i^j at L_i. For this case, since each level at level L_{i+1} there may be O(n/ls) intervals which fall within the range determined by ID_i^j. Hence, an O(log(n/ls))=O(log n) search may be needed at every level.

Consequently, virtual intervals at levels $L_i \leqq L_{lsMAX}$ are defined in the following manner. The number of intervals N_i is defined at level L_i. Boundary points that demarcate the N_i intervals in the d dimension at level L_i are denoted by y_1^i, y_2^i . . . with a maximum of $2N_i$ such points. Every other point at level L_i is replicated at level L_{i-1}, and up to $2N_i$ points are so propagated to level L_{i-1}. Although the present embodiment is described using propagation of every other point, other embodiments may skip NS points, NS an integer greater than 1, or may vary the number of points skipped according to granularity of the pointers used.

The points that were propagated together with the points defining original non-overlapping intervals ID_i^j, now define intervals at level L_{i-1} as new intervals VD_{i-1}^j. These intervals are stored as non-overlapping intervals at level L_{i-1}. Next, for all the intervals at level L_{i-1} and their associated points, every other point is replicated and propagated as virtual points to level L_{i-2}. This propagation process is repeated until the root level L_{ie}, (i.e., L_1) is reached. Note that the propagation process is employed to speed up the search; at each level, the filter-rule rectangles associated with each non-overlapping interval are as described in the preprocessing algorithm described previously. Virtual intervals and points that result from propagation are desirably ignored for association of filter-rule rectangles with non-overlapping intervals.

The propagation process increases memory space requirements by a constant factor, and so the total memory space requirement is still O(n). A maximum amount of virtual intervals created and corresponding maximum memory space is when N_{lsMAX}=n, n being the number of filter rules, in which case the number of boundary points at level L_{sMAX} is 2n. The extra memory space due to the propagations is then as given in equation (1)

$$\left(n+\frac{n}{2}+\frac{n}{4}+\ldots\right)\leq 2n \tag{1}$$

Increasing the memory space by a constant factor, however, allows for searching of multiple lists (i.e. lists of non-overlapping intervals at each level) efficiently. A packet EP=(S, D) arrives at the packet filter and is processed by the classification algorithm with a filter-rule table organized in accordance with the alternative embodiment. A first level, i.e., L_1 list of non-overlapping intervals VD_1^j is searched as described previously with respect to the classification algorithm, taking O(log n) time for the worst case. This search results in locating the given point D in an interval VD_1^j that may be a virtual interval propagated from the level L_2. With D localized to this interval ID_1^j, a search in the next level L_2searches in the range of intervals given by VD_1^j. Because every other point has been propagated up from level L_2, only 2 intervals in VD_2^j may fall within the interval VD_1^j to which D has been localized. Hence, the search at level L_2 may be completed in O(1) time. In general, in moving from level L_i to level L_{i+1}, the propagation of intervals allows enough information gained in the search at level L_i to be employed in the search at level L_{i+1} is O(1) time. Hence, the worst case execution time of the look-up algorithm of the alternative embodiment is $O(l_{sMAX}+\log n)$.

FIG. 9A and 9B illustrate an example of an alternative embodiment of the packet filter employing virtual intervals to reduce search time of the classification algorithm. FIG. 9A illustrates a trie structure employed to search prefix values of fourteen exemplary filter rules in ascending order of length. FIG. 9B shows creation of virtual intervals for levels of a portion of the trie structure shown in FIG. 9B. For the exemplary embodiment of FIG. 9A and FIG. 9B, Table 1 provides a list of filter-rules with corresponding prefix values and lengths for source fields and destination field ranges.

TABLE 1

Filter-Rule Number	Source Prefix Value	Prefix length	Destination range d (lower bound, upper bound)
	11*	2	(0,15)
2	0*	1	(4,7)
3	00*	2	(12,15)
4	0*	1	(12,15)
5	0*	1	(8,15)
6	10*	2	(8,15)
7	001*	3	(8,15)
8	000*	3	(6,7)
9	000*	3	(4,5)
10	001*	3	(8,9)
11	001*	3	(4,5)
12	001*	3	(10,11)
13	001*	3	(12,13)
14	001*	3	(0,3)

A packet EP with fields S=0010 and D=1101 arrives in the system. Referring to FIG. 9A, a search of the trie structure 900 (the trie search) in the s-dimension begins at the root level 901 (level 0) to determine if the source address (S=0xxx) begins with a 0(state 902) or a 1 (state 903). This is a search of the set of prefixes of length 1. The trie search

moves to the state **902** at level **1** corresponding to the prefix 0xxx of length **1**. Similarly, at level **2** the trie search determines if the next bit of the source address (S=00xx) is a 0 (state **904**) or a 1 (state **905**). The trie search moves to the state **904** at level **2** corresponding to the prefix 00xx of length **2**. Finally, at level 3 the trie search of a portion of the set of prefixes of length 3 determines if the next bit of the source address (S=001x) is a 0 (state **908**) or a 1 (state **909**). The trie search moves to the state **909** at level **3** corresponding to the prefix 001x of length **3**. For searches of prefixes, only a portion of sets of prefixes are searched in the tries. Consequently, states **903**, **906** and **907** are not reached since the trie search moves from state **901** to state **902**, to state **904**.

FIG. **9B** illustrates an example of virtual intervals and point propagation to reduce search time of the classification algorithm. First, non-overlapping intervals in the d-dimension are shown for selected states at each level. For example, at level 1, state **902** corresponds to the prefix of length **1** being 0xxx. The filter-rules of this prefix 0xxx (from Table 1) are rules 2, 4 and 5 with respective filter-rule segments (decimal ranges in the d-dimension) of **(4,7)**, **(8,12)** and **(8,15)**. These filter-rule segments are then decomposed into non-overlapping intervals **(4,7)**, **(8,12)** and **(12, 15)**. Without virtual intervals, the trie search at level **1** searches these three intervals to find the value D=1101 (i.e., 13 decimal) included in the third non-overlapping interval **(12,15)** associated with rule **5**. However, for the next level **2**, the information of this search is lost.

Referring to FIG. **9B**, the non-overlapping intervals of the highest level, level **3**, are shown for the states **908** and **909**. Points of these original, non-overlapping intervals at level **3** are propagated to the non-overlapping intervals at level **2**. Brackets in FIG. **9B** indicate original, non-overlapping intervals. For the example shown, alternate points of the intervals of the left state **908** (next bit **0**) and right state **909** (next bit **1**) are inserted into the non-overlapping intervals of the states of the next level **2**, but as described previously the present invention is not so limited. For example, virtual intervals **(0,3)**, **(3,4)**, **(5,6)**, **(6,9)**, **(9,11)**, **(11,12)**, **(12,13)** and **(13,15)** are created from the original non-overlapping interval **(12,15)**. Next, the alternate points of the intervals of state **904** are propagated to level **1**, and as shown, propagated points, such as **12**, may be duplicated in a level, since pointers are to be associated with the intervals. Normally, points of left and right states are propagated, but for the example of FIG. **9A** and FIG. **9B**, no rules or intervals are associated with state **905**.

As the trie search of prefixes as shown in FIG. **9A** progresses, the search of intervals is as shown in FIG. **9B**. At level **1**, state **902**, the intervals in the d-dimension are searched and the value of D=1101, 13 decimal, is determined to be included in the interval **(12, 12,15)**. At level **2**, after the prefix search moves to state **904**, the pointer associated with propagated point **12** in interval **(12,12,15)** is employed to limit the search in level **2** to interval **(12,13,15)**. At level **3**, after the prefix search moves to state **909**, the pointer associated with propagated point **13** in interval **(12,13,15)** is employed to limit the search in level **3** to interval **(12,13)**, associated with rule **13** of Table 1.

As described, the algorithm for computing the filters is largely implemented in hardware and may be manufactured in application specific integrated circuit (ASIC) form, or as a field programmable gate array (FPGA) that consequently, may operate at very high speed. FIG. **10** illustrates the hardware system **1000** for implementation of the packet filter in accordance with the present invention in a packet forwarding engine or router, including an input line **1005** for receiving an incoming packet and a bi-directional CPU interface line **1010** representing control and timing lines for purposes of illustration. The incoming packet is input to a pipeline register **1025** for temporary storage and is also input to each classification processor **1050**. Classification processor **1050** employs memory **1030** to identify a filter-rule to be applied to the incoming packet. Field processor **1035** updates fields of the packet stored in pipeline register **1025** based on the identified filter-rule to be applied to the incoming packet. The details of classification processor **1050** are now described with reference to FIG. **11**.

FIG. **11** shows a classification processor **1050** that receives the incoming packet and stores field parameters, e.g., source address and destination addresses S and D, in a register **1176**. Under the control of filter processor **1160**, optional memory control device **1165**, and associated memory **1030**, the search of the classification algorithm is performed whereby non-overlapping interval information from memory **1030** is provided to the register **1179** for each prefix length. Comparator **1180** performs a comparison to ascertain each interval associated with the D value of the received packet. After the correct solution for a filter-rule rectangle is found, its corresponding bitmap vector containing potential filter-rule actions is provided from register **1179** along line **1190**. From the resultant bitmap vector, the CPU will apply the rule of highest priority, and performs the action dictated by the filter rule upon the received packet stored in the pipeline register **1025**. Thus, the packet may be dropped or forwarded to another destination on output line **1015**.

The preprocessing algorithm of the present invention may be implemented in the classification processor by filter-rule processing and table processing modules. The filter-rule processing module may assign filter-rules to prefix values and lengths in one dimension, project the filter-rule segments in the other dimension, and decompose the filter-rule segments into non-overlapping intervals. The table-processing module may be employed to coordinate memory organization and storage, generating the necessary pointers with non-overlapping intervals for particular prefix value addressing schemes.

An example memory organization for the system is illustrated in FIG. **12**, which depicts a filter-rule table having a plurality of interval lists in one dimension corresponding to each prefix length of another dimension, which may be associated with the following respective filter parameters: 1) destination addresses, and 2) source address. Entries of the filter-rule table are generated as described previously, i.e., with respect to FIGS. **6** and **7**, and addressed by prefix values **1259***a*–**1259***d*. Each filter-rule table is shown to include an array **1260***a*–**1260***d* of intervals to be searched corresponding to prefix values as described above with reference to FIG. **8**, and the corresponding filter actions **1261***a*–**1261***d* and the pointers **1262***a*–**1262** *d*.

While embodiments of the present invention are shown and described with respect to searches in a given dimension ordered from shortest to longest length, as would be apparent to one skilled in the art the present search algorithms and/or filter-rule table structures may be varied. For example, the search may be from the longest to the shortest prefix length, or from initial to final prefix values in an ordered list of the set of prefix values. Further, matching of packets field values with prefix values and interval values are described herein using binary search techniques, but the present invention is not so limited. As would be apparent to one skilled in the art, other search techniques to match values may be employed, such as employing a perfect hash method.

It will be further understood that various changes in the details, materials, and arrangements of the parts which have been described and illustrated in order to explain the nature of this invention may be made by those skilled in the art without departing from the principle and scope of the invention as expressed in the following claims.

What is claimed is:

1. Apparatus for associating at least one filter rule with a packet, each filter rule and the packet characterized by values in first and second dimensions, the filter rule to be applied to the packet by a router in a communications network, the apparatus comprising:

a storage medium adapted to store a filter-rule table, each entry of the filter-rule table corresponding to a prefix value having a length in the first dimension and at least one interval in the second dimension; and

a classification processor comprising:

a comparator adapted to identify each prefix value matching the value of the packet in the first dimension, and

a filter processor adapted to retrieve, from the filter-rule table, each interval associated with each prefix value identified by the comparator containing the value of the packet in the second dimension,

wherein the filter processor identifies as a solution interval the interval associated with the prefix length characterized by an associated predetermined metric and containing the second field, and

wherein the classification processor associates the filter rule corresponding to the solution interval with the packet.

2. The invention as recited in claim 1, wherein the classification processor further comprises a pre-processor including:

a filter-rule processing module adapted to:

assign each filter-rule to one or more prefix values based on the values in the first dimension,

project, for each prefix value having the same length, values of each corresponding filter rule of the prefix value onto the second dimension to define at least one filter-rule segment, and

decompose each filter-rule segment into one or more non-overlapping intervals associated with each prefix value of the same length in the second dimension; and

a table-processing module adapted to generate a pointer for each corresponding non-overlapping interval to identify an included filter-rule, the table-processing module adapted to store the pointer as an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval.

3. The invention as recited in claim 2, wherein:

the filter-rule processing module further comprises:

assigning means for assigning each prefix value of the same length to a corresponding level;

first projecting means for projecting, for the level having prefix values of a first length, values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment;

second projecting means for projecting, in each level beginning at the level having prefix values having a second length, 1) values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment in a current level, and 2) selected points of the at least one non-overlapping interval in the previous level so as to define at least one virtual interval in the second dimension; and

interval forming means for forming each filter-rule segment and each virtual interval of the current level into one or more non-overlapping intervals associated with each prefix value having the same length.

4. The invention as recited in claim 3, wherein the first and second lengths are either 1) the longest and next longest lengths in a descending prefix length order, respectively, or 2) the shortest and next shortest lengths in an ascending prefix length order, respectively.

5. The invention as recited in claim 3, wherein the second projecting means projects, as selected points, every Nth point that defines either a start point or a stop point of each non-overlapping interval in the previous level, N an integer greater than 1.

6. The invention as recited in claim 2, wherein the values of each filter rule in the second dimension are at least one range being a power of 2, each range being projected as a corresponding filter-rule segment to form the non-overlapping interval in the second dimension.

7. The invention as recited in claim 1, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

8. The invention as recited in claim 1, wherein an entry of the filter-rule table of the storage medium includes a pointer identifying at least one filter rule contained in the corresponding non-overlapping overlapping interval.

9. The invention as recited in claim 8, wherein each filter-rule has an associated priority, and the pointer identifies the filter-rule with the highest associated priority contained in the corresponding non-overlapping interval.

10. The invention as recited in claim 8, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

11. The method as recited in claim 1, wherein the associated predetermined metric is either the prefix value having the longest prefix length, the shortest prefix length or the prefix length having a highest priority.

12. A method of associating at least one filter rule with a packet, each filter rule and the packet characterized by values in first and second dimensions, the filter rule to be applied to the packet by a router in a communications network, the method comprising the steps of:

a) providing a filter-rule table, each entry of the filter-rule table corresponding to a prefix value having a length in the first dimension and at least one interval in the second dimension;

b) identifying each prefix value matching the value of the packet in the first dimension;

c) retrieving, from the filter-rule table, each interval associated with each prefix value identified in step b) containing the value of the packet in the second dimension;

d) identifying, as a solution interval, the interval associated with the prefix value characterized by an associated predetermined metric and containing the value of the packet in the second dimension; and

e) associating the filter rule corresponding to the solution interval with the packet.

13. The method as recited in claim 12, wherein the step a) comprises the steps of:

f) assigning each filter-rule to one or more prefix values based on the values in the first dimension;

g) projecting, for each prefix value having the same length, values of each corresponding filter rule of the prefix value onto the second dimension to define at least one filter-rule segment;

h) decomposing each filter-rule segment into one or more non-overlapping intervals associated with each prefix value of the same length in the second dimension;

i) generating a pointer for each corresponding non-overlapping interval to identify an included filter-rule; and

j) storing the pointer as an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval.

14. The method as recited in claim **13**, wherein:

step g) further comprises the steps of:

g1) assigning each prefix value of the same length to a corresponding level;

g2) projecting, for the level having prefix values having a first length, values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment,

g3) projecting, in each level beginning at the level having prefix values having a second length, 1) values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment in a current level, and 2) selected points of the at least one non-overlapping interval in the previous level so as to define at least one virtual interval in the second dimension; and

step h) further comprises the step of:

h1) forming each filter-rule segment and each virtual interval of the current level into one or more non-overlapping intervals associated with each prefix value having the same length.

15. The method as recited in claim **14**, wherein, for steps g2) and g3), the first and second lengths are either 1) the longest and next longest lengths in a descending prefix length order, respectively, or 2) the shortest and next shortest lengths in an ascending-prefix length order, respectively.

16. The method as recited in claim **14**, wherein step g3) projects, as selected points, every Nth point that defines either a start point or a stop point of each corresponding non-overlapping interval in the previous level, N an integer greater than 1.

17. The method as recited in claim **13**, wherein the values of each filter rule in the second dimension are at least one range being a power of 2, the projecting step g) projects each range as a corresponding filter-rule segment in the second dimension, and the decomposing step h) forms the non-overlapping interval from the corresponding filter-rule segment projected in step g).

18. The method as recited in claim **12**, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

19. The method as recited in claim **12**, wherein, for the filter-rule table provided in step a), an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval includes a pointer identifying at least one filter rule contained in the corresponding non-overlapping interval.

20. The method as recited in claim **19**, wherein each filter-rule has an associated priority, and the pointer generated in step i) identifies the filter-rule with the highest associated priority contained in the corresponding non-overlapping interval.

21. The method as recited in claim **19**, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

22. The method as recited in claim **12**, wherein for step d) the associated predetermined metric is either the prefix value having the longest prefix length, the shortest prefix length or the prefix length having a highest priority.

23. A method of storing at least one filter rule with values associated with first and second dimensions in a filter-rule table comprising the steps of:

a) assigning each filter-rule to one or more prefix lengths based on the values in the first dimension;

b) projecting, for each prefix length, values of each corresponding filter rule of the prefix length onto the second dimension to define at least one filter-rule segment,

c) decomposing each filter-rule segment into one or more non-overlapping intervals associated with each prefix length and corresponding filter rule in the second dimension;

d) generating a pointer for each corresponding non-overlapping interval to identify an included filter-rule; and

e) storing the pointer as an entry of the filter-rule table associated with a prefix length and a non-overlapping interval.

24. The method as recited in claim **23**, wherein:

step b) further comprises the steps of:

b1) assigning each prefix value of the same length to a corresponding level;

b2) projecting, for the level having prefix values of a first length, values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment,

b3) projecting, in each level beginning at the level having prefix values having a second length, i) values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment in a current level, and ii) selected points of the at least one non-overlapping interval in the previous level so as to define at least one virtual interval in the second dimension; and

step c) further comprises the step of:

c1) forming each filter-rule segment and each virtual interval of the current level into one or more non-overlapping intervals associated with each prefix value having the same length.

25. The method as recited in claim **24**, wherein, for steps b2) and b3), the first and second lengths are either 1) the longest and next longest lengths in a descending prefix length order, respectively, or 2) the shortest and next shortest lengths in an ascending prefix length order, respectively.

26. The method as recited in claim **24**, wherein step b3) projects, as selected points, every Nth point that defines either a start point or a stop point of each corresponding non-overlapping interval in the previous level.

27. The method as recited in claim **23**, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

28. The method as recited in claim **23**, wherein each pointer stored in the filter-rule table in step e) identifies each filter rule contained in the non-overlapping interval.

29. The method as recited in claim **23**, wherein each pointer stored in the filter-rule table in step e) identifies the filter-rule with the highest associated priority contained in the corresponding non-overlapping interval.

30. The method as recited in claim **23**, wherein the values of each filter rule in the second dimension are at least one range being a power of 2, the projecting step b) projects each range as a corresponding filter-rule segment in the second dimension, and the decomposing step c) forms the non-overlapping interval from the corresponding filter-rule segment projected in step b).

* * * * *